SISTEM DETEKSI DAN PENGENALAN JENIS RAMBU LALU LINTAS MENGGUNAKAN METODE *SHAPE DETECTION* PADA RASPBERRY PI

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Nama: Olivia Rumiris Sitanggang
NIM: 145150301111003

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

SISTEM DETEKSI DAN PENGENALAN JENIS RAMBU LALU LINTAS MENGGUNAKAN METODE *SHAPE DETECTION* PADA RASPBERRY PI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Nama: Olivia Rumiris Sitanggang
NIM: 145150301111003

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Hurriyatul Fitriyah, S.T., M.Sc	Dr. Eng. Fitri Utaminingrum, S.T., M.T
NIP:19851001 201504 2 003	NIP:19820710 200812 2 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 27 Juli 2018

Olivia Rumiris Sitanggang

NIM: 145150301111003

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yesus Kristus atas segala berkatNya sepanjang penelitian ini sehingga penulis dapat menyelesaikan skripsi yang berjudul "Sistem Deteksi dan Pengenalan Jenis Rambu Lalu Lintas Menggunakan Metode *Shape Detection* pada Raspberry Pi". Penelitian sudah menghabiskan waktu kurang lebih satu tahun. Selama proses pelaksanaanya, peneliti banyak menghadapi kendala pengerjaan dan tantangan pengambilan data. Namun dengan bantuan, dukungan, dan motivasi dari banyak pihak, maka penulis dapat menyelesaikan penelitian ini hingga akhir. Oleh karena itu, melalui kesempatan ini penulis ingin menyampaikan terimakasih sebesar-besarnya kepada:

1. Ibu Hurriyatul Fitriyah, S.T., M.Sc selaku dosen pembimbing I yang telah memberikan penulis waktu yang banyak, tenaga, ilmu, dan motivasi yang tulus kepada penulis sehingga dapat menyelesaikan penelitian ini.
2. Ibu Dr. Eng. Fitri Utaminingrum, S.T., M.T selaku dosen pembimbing II yang sudah memberi waktu yang banyak, membimbing penulis dari dasar, motivasi yang tulus, dan selalu semangat mengingatkan penulis.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika.
4. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya Malang.
5. Kedua orang tua penulis, Bapak Thomson Sitanggang dan Ibu Freeyka Simangunsong yang selalu berdoa dan memberikan semangat dan dukungan. Jonathan Sitanggang selaku saudara penulis yang telah mendukung dan selalu mengingatkan penulis sepanjang penelitian ini.
6. Seluruh civitas akademika Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi Informatika di Universitas Brawijaya dan selama penyelesaian skripsi ini.
7. Thomas Oddy selaku rekan penulis yang telah mendukung penulis dari awal dan hingga akhir penelitian ini.
8. Rekan-rekan penulis Muliyahati, Cindy Lilian, Bramantyo Ardi, Enno Roscitra, Ingrid Melanika, Dinda Agnes, Faizal Andy, Linda Silvya yang telah membantu proses pengujian penelitian.
9. Leo Justicia selaku sahabat penulis yang selalu bersedia mendengar, mendoakan, membimbing, dan memberikan dorongan kepada penulis.
10. Sahabat "Hygge" Muliyahati, Enno Roscitra, Yohana Kristinawati, Bunga Boru, Riyyan Royhan yang selalu mendoakan dan membantu penulis hingga akhir penelitian ini.

11. Keluarga KTB yang telah mendoakan, memberi semangat dan menemani penulis hingga akhir penelitian ini.
12. Keluarga Besar Teknik Komputer yang selalu memberi semangat dan dukungan.
13. Keluarga Besar Laboratorium Sistem Komputer dan Robotika yang telah mendukung penulis.

Hingga penelitian ini selesai, penulis masih merasa memiliki banyak kekurangan. Maka dari itu, penulis mengharapkan saran dan kritik dari pembaca untuk membangun pengembangan berikutnya dari penelitian ini.

Malang, 27 Juli 2018

Penulis

oliviasitanggangr@gmail.com

ABSTRAK

Pengenalan Rambu Lalu Lintas (*Traffic Sign Recognition*) adalah teknologi pengolahan citra digital yang digunakan untuk mengenali rambu secara *real-time*. Teknologi ini diterapkan dalam *Driver Assistance System* (DAS). Pengenalan Rambu Lalu Lintas terdiri dari dua fase utama, yaitu deteksi rambu dan pengenalan. Deteksi merupakan fase untuk menemukan kemungkinan area gambar dimana rambu berada. Keluaran dari proses deteksi adalah gambar hasil segmentasi berisi *Region of Interest* (ROIs) yang dapat mengenali daerah potensial rambu jalan berada. Daerah potensial tersebut akan mempengaruhi *input* proses pengenalan (*recognition*). Sehingga dibangun sebuah sistem pendeteksi dan pengenalan jenis rambu. Sistem ini diimplementasikan pada Raspberry Pi dan bekerja *real-time* saat mengolah citra dari rambu dari *webcam*. Algoritma deteksi dibagi menjadi tiga bagian utama, yaitu segmentasi warna, *shape detection*, dan klasifikasi rambu. Metode yang diterapkan dalam penelitian ini adalah metode pengenalan bentuk. Metode ini didukung oleh jumlah titik dari objek sebegai representasi dari tiap-tiap bentuk dan perbandingan luas objek kontur dengan *bounding rectangle*. *Output* dari sistem ini berupa notifikasi suara jenis rambu bagi pengendara. Diharapkan dengan metode ini proses pendeteksian untuk menemukan regional rambu lebih akurat sehingga memperkecil jumlah informasi yang perlu diolah dalam tahap pengenalan. Tingkat keberhasilan deteksi jenis rambu perintah, larangan, dan rambu peringatan yaitu 80.7%, hasil pengujian warna dari ketiga rambu mencapai angka 85.45%, dan hasil persentase mengenali bentuk dari rambu adalah 80.7%. durasi mendeteksi jenis rambu ini menghasbiskan 0.5 detik untuk satu frame atau 2 frame perdetiknya dengan jarak deteksi 2-5 meter.

Kata kunci: rambu lalu lintas, *shape detection*, deteksi

ABSTRACT

The traffic sign recognition is the digital image processing technology that used to recognize the sign in real-time. This technology applied in the Driver Assistance System. The road sign recognition consist of 2 main phase, they are road detection and recognition. Detection is the phase to find the possibility of picture area where the sign is located. The output from the detection process is the result picture segmentation that contain region of interest that can recognize the potential area of where the road sign being located. Those potential area will be affected the input of recognition process. So built a system of detection and recognition of the type of signs. This system is implemented on raspberry pi and real-time when processing the image of road sign from webcacm. The detection of algorithm consist into three main part, they are color segmentation, shape detection, and road classification. The method that being applied in this research is shape recognition method. This method is supported by the amount of point from the object as a representation of the amount of side from every shape and the comparison of object area with the bounding rectangle. And the output of this system is a kind of the sign notification for drivers. It is expected with this method the detection process to find the accurate regional sign recognition. The level of success in detecting kind of command signs, prohibition, and warning sign are 80.7%, the result of color examination from the three signs reach the number of 85.45%, and the result of presentation in recognizing the shape of sign is 80.7%. the duration of detecting of traffic signals is 0.5 seconds (for each frame) or 2 frames per second with detection distance 2-5 meters.

Keywords : traffic sign, shape detection, detection

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Pengenalan rambu lalu lintas (*Traffic Sign Recognition*) adalah teknologi yang digunakan untuk mengenali rambu jalan secara *real-time*. Teknologi ini diterapkan dalam *Driver Assistance System* (DAS). Selain pengenalan rambu jalan ada beberapa teknologi dalam sistem *Driver Assistance* diantaranya deteksi garis *lane* jalan, bantuan parkir, monitoring gangguan pengemudi, kemudi otomatis atau sering disebut dengan *auto pilot*, dan yang saat ini sudah banyak digunakan GPS (*Global Positioning System*). Pengenalan rambu berperan penting untuk membantu pengemudi yang melewatkan dan tidak mengetahui keberadaan rambu sebagai visualisasi perubahan situasi jalan saat itu. Kelalaian ini bersifat membahayakan hingga dapat menyebabkan kecelakaan (Ilmuti.org).

Pengenalan rambu lalu lintas terdiri dari dua fase utama, yaitu deteksi rambu dan pengenalan. Deteksi merupakan fase untuk menemukan kemungkinan area gambar dimana rambu berada. Keluaran dari proses deteksi adalah gambar hasil segmentasi yang berisi *Region of Interest* (ROIs) yang dapat mengenali daerah potensial rambu jalan berada. Daerah potensial tersebut akan menjadi *input* proses pengenalan (*recognition*). Sedangkan fase pengenalan berfungsi mengidentifikasi tujuan dari masing-masing rambu dengan membandingkan informasi dari fase sebelumnya atau model rambu jalan. Model ini rata-rata diperoleh berdasarkan fitur masing-masing rambu maupun *template* rambu dalam data set (I. Sebanja, 2010).

Hasil penelitian (S.S.M. Sallah, 2014) *Road Sign Detection and Recognition System for Real-Time Embedded Applications* dibangun sebuah sistem untuk deteksi dan pengenalan rambu dengan akurasi 90.7%. sistem ini dibagi kedalam dua bagian yaitu deteksi dan pengenalan. Deteksi rambu terdiri dari tiga tahap yaitu segmentasi warna, pengenalan bentuk, dan klasifikasi rambu. Setelah melewati tiga tahap tersebut maka diperoleh daerah rambu yang diberi *bounding box* sebagai tanda rambu ditemukan. Informasi rambu tersebut menjadi masukan ke tahap pengenalan, tahap ini dilakukan dengan perbandingan luas dan keliling *region of interest*.

Deteksi dan pengenalan terbagi menjadi tiga modul utama yaitu *pre-processing*, *detection*, dan *recognition*. *Pre-processing* menghilangkan derau pada gambar dan mempertinggi kualitas gambar. Proses deteksi melakukan segmentasi berdasarkan warna pada gambar dengan keluaran berupa *region of interest* gambar yang kemungkinan sebuah rambu. kemungkinan ini menjadi input pada tahap *recognition*. Persentase hasil deteksi penelitian oleh (Sheikh, 2016) yang berjudul *Traffic Sign Detection and Classification using Colour Feature and Neural Network* mencapai 90% dan hasil pengenalan lebih dari 88%.

Rambu lalu lintas memiliki warna dan bentuk yang berbeda-beda. Beberapa objek memiliki warna yang sama dengan warna rambu sehingga pengenalan bentuk perludilakukan. Maka dibangun sebuah sistem deteksi dan pengenalan

pada jenis rambu lalu lintas menggunakan metode *shape detection*. Algoritma deteksi dibagi menjadi tiga bagian utama, yaitu segmentasi warna, *shape detection*, dan klasifikasi rambu. Metode ini didukung oleh jumlah titik dari objek sebegai representasi jumlah titik puncak dari tiap-tiap bentuk dan perbandingan luas objek dengan *bounding rectangle*. Sistem ini bekerja didalam Raspberry Pi dan *real-time* saat mengolah citra dari rambu jalan. *Output* dari sistem ini berupa notifikasi suara dari jenis rambu bagi pengendara. Diharapkan dengan metode ini proses pendeteksian untuk menemukan regional rambu lebih akurat sehingga memperkecil jumlah informasi yang perlu diolah dalam tahap pengenalan (OpenCV, n.d.).

1.2 Rumusan masalah

Berdasarkan latar belakang di atas dapat dirumuskan masalah penelitian sebagai berikut:

1. Bagaimana merancang sistem deteksi dan pengenalan jenis rambu lalu lintas menggunakan pemrosesan citra digital pada Raspberry Pi?
2. Bagaimana cara mengenali rambu perintah, peringatan, dan rambu larangan dengan metode *shape detection*?
3. Bagaimana nilai akurasi sistem deteksi warna, bentuk, dan jenis rambu lalu lintas (perintah, peringatan, dan larangan) pada Raspberry Pi?
4. Bagaimana rata-rata waktu yang dibutuhkan untuk mendeteksi rambu lalu lintas (perintah, peringatan, dan larangan)?

1.3 Tujuan

Berikut beberapa tujuan dalam penelitian ini yaitu:

1. Merancang sistem deteksi dan pengenalan jenis rambu lalu lintas menggunakan pemrosesan citra digital pada Raspberry Pi.
2. Mengenali ketiga jenis rambu menggunakan metode *shape detection*.
3. Mendapatkan tingkat keberhasilan sistem deteksi dari jenis, warna, dan bentuk rambu lalu lintas (perintah, peringatan, dan larangan) pada Raspberry Pi.
4. Mendapatkan nilai rata-rata waktu deteksi rambu lalu lintas (perintah, peringatan, dan larangan).

1.4 Manfaat

1. Membantu pengembangan teknologi pengenalan rambu lalu lintas pada *Driver Assistance System*.
2. Mengurangi tingkat kecelakaan yang saat ini sangat tinggi di Indonesia.
3. Menyediakan referensi untuk membantu peneliti selanjutnya di penelitian pengenalan rambu lalu lintas.

1.5 Batasan masalah

1. Rambu lalu lintas yang digunakan hanya rambu yang dikeluarkan oleh Kementerian Perhubungan Indonesia.
2. Rambu lalu lintas yang digunakan adalah rambu perintah, rambu peringatan, dan larangan.
3. Rambu yang digunakan sebagai objek penelitian terdiri dari 3 warna, yaitu merah, biru, dan kuning.
4. Rambu yang digunakan sebagai objek penelitian terdiri dari 4 bentuk yaitu bulat (*circle*), segitiga (*triangle*), oktagonal, dan kotak (*square/diamond*).
5. Rambu lalu lintas dalam kondisi tidak rusak dan tidak diberi stiker.
6. Resolusi kamera yang digunakan adalah 1280x720 piksel.
7. Kendaraan berjalan pada kecepatan 30km/jam.
8. Sistem dapat bekerja pada pagi, siang, dan sore hari dalam kondisi cuaca tidak hujan.

1.6 Sistematika pembahasan

Sistematika pembahasan penelitian ini akan disusun seperti pada bab-bab yang akan diuraikan dibawah ini:

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan penelitian, dan sistematika pembahasan "SISTEM DETEKSI DAN PENGENALAN JENIS RAMBU LALU LINTAS MENGGUNAKAN METODE *SHAPE DETECTION* PADA RASPBERRY PI".

BAB II LANDASAN KEPUSTAKAAN

Bab ini mengulas landasan teori dan kajian pustaka dari penelitian terdahulu sebagai dasar penelitian ini. Teori-teori pendukung penelitian ini antara lain rambu lalu lintas, pengolahan citra digital, konversi RGB ke HSV, *thresholding*, kontur, *shape detection*, OpenCV, dan *Hardware*.

BAB III METODE PENELITIAN

Bab ini mengkaji mengenai langkah kerja dalam penelitian ini seperti kajian literatur, analisis kebutuhan, perancangan sistem, implementasi sistem, serta pengujian dan analisis.

BAB IV ANALISIS KEBUTUHAN

Bab rekayasa kebutuhan memaparkan kebutuhan sistem yang harus dipenuhi untuk kebutuhan perancangan dan implementasi sistem. Beberapa kebutuhan tersebut adalah kebutuhan perangkat keras, kebutuhan fungsional, dan kebutuhan non-fungsional dan gambaran umun sistem.

BAB V PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan perancangan dan implementasi dari algoritma deteksi rambu dari penelitian "SISTEM DETEKSI DAN PENGENALAN JENIS RAMBU LALU LINTAS

MENGGUNAKAN METODE *SHAPE DETECTION* PADA RASPBERRY PI" dengan rinci dan beruntun.

BAB V PENGUJIAN DAN ANALISIS

Bab ini menguraikan proses uji coba sistem yang dibangun dan menganalisis *output* yang di hasilkan oleh sistem "SISTEM DETEKSI DAN PENGENALAN JENIS RAMBU LALU LINTAS MENGGUNAKAN METODE *SHAPE DETECTION* PADA RASPBERRY PI".

BAB VI KESIMPULAN DAN SARAN

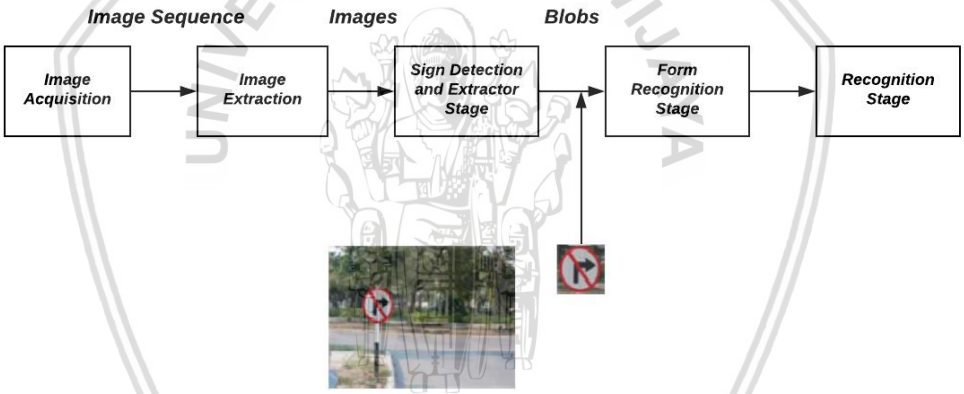
Bab ini menguraikan kesimpulan dari hasil pengujian dan analisis yang dilakukan pada sistem "SISTEM DETEKSI DAN PENGENALAN JENIS RAMBU LALU LINTAS MENGGUNAKAN METODE *SHAPE DETECTION* PADA RASPBERRY PI" dan saran pengembangan untuk penelitian berikutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menguraikan referensi yang berasal dari penelitian terdahulu terkait dengan penelitian saat ini dan dasar-dasar teori yang melandasi penelitian "SISTEM DETEKSI DAN PENGENALAN JENIS RAMBU LALU LINTAS MENGGUNAKAN METODE *SHAPE DETECTION* PADA RASPBERRY PI".

2.1 Landasan Pustaka

Kajian pustaka yang pertama adalah penelitian dari (P. Shopa, 2015) yang berjudul *Traffic Sign Detection and Recognition Using OpenCV*. Sama hal nya dengan penelitian penulis, penelitian ini bekerja dengan *real-time*, menggunakan HSV sebagai filter warna rambu, menemukan objek rambu dengan proses *threshold* dan kontur, dan menggunakan pustaka OpenCV. Yang menjadi perbedaan diantara keduanya adalah bahwa penelitian ini dilakukan hingga tahap pengenalan rambu seperti pada Gambar 2.1. Namun penelitian ini hanya mengenali warna rambu merah dan hanya dalam dua bentuk saja yaitu bulat dan oktagonal.

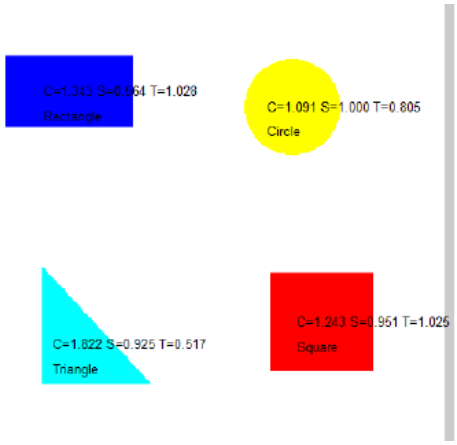


Gambar 2.1 Blok Diagram Sistem Deteksi dan Pengenalan Rambu

(Sumber : P. Shopa, 2015)

Sistem deteksi warna dan dan bentuk geometrik dengan pengolahan citra oleh (Chhaya, 2015) bekerja pada aplikasi MATLAB. Persamaan diantara penelitian ini dan penulis yang pertama adalah pengenalan bentuk menggunakan bantuan bounding rectangle dan menghitung perbandingan luas pada objek. Dari hasil pengujian pada objek diperoleh akurasi sebesar 95%. *Background* yang digunakan pada penelitian ini hanya *background* hitam dan sederhana sehingga hanya memerlukan fungsi *grayscale*. k

Sebagai pengembangan lanjut dari kedua penelitian terdahulu, maka dibentuk sistem deteksi dan pengenalan jenis rambu pada pustaka OpenCV berbasis *Python*. Sistem ini bekerja didalam sebuah mikro komputer yaitu Raspberry Pi. Jumlah warna yang dideteksi ada 3 (merah, kuning, biru) yang terdiri dari 4 bentuk (*circle*, *triangle*, oktagonal, dan *square*). Data yang diolah berupa video dari rekaman jalan oleh *webcam* sehingga *background* yang digunakan kompleks.



Gambar 2.2 Hasil akhir sistem

(Sumber : Chhaya, 2015)

2.2 Dasar Teori

Pada sub bab ini membahas beberapa teori yang mendukung penelitian, yaitu: pengolahan citra digital, konversi RGB ke HSV, *thresholding*, kontur, *shape detection*, rambu lalu lintas, kebutuhan *hardware*, OpenCV.

2.2.1 Rambu Lalu Lintas

Rambu lalu lintas adalah salah satu sarana prasarana jalan bagi pengendara maupun pengguna jalan. Rambu lalu lintas dapat diimplementasikan kedalam bentuk lambang, huruf, angka, kalimat dan perpaduan di antaranya yang berfungsi sebagai peringatan, larangan, perintah dan petunjuk bagi pemakai jalan (Kementerian Perhubungan, 2014).

2.2.1.1 Rambu Larangan

Beberapa rambu berfungsi untuk memberi informasi maupun perintah namun rambu larangan menghimbau agar pengemudi atau pengguna jalan menghindari larangan-larangan di jalan. Rambu-rambu ini dilengkapi dengan latar putih atau merah seperti pada Gambar 2.3.

Gambar 2.3 Rambu Larangan

(Sumber : Kementerian Perhubungan, 2014)

2.2.1.2 Rambu Perintah

Rambu Perintah berfungsi untuk memberi petunjuk atau keterangan kepada pengemudi atau pengguna jalan, tentang informasi lajur mengemudi. Rambu ini biasanya dalam bentuk bundar dan berwarna biru seperti pada Gambar 2.4.

Gambar 2.4 Rambu Perintah

(Sumber : Kementerian Perhubungan, 2014)

2.2.1.3 Rambu Peringatan

Rambu peringatan merupakan rambu keterangan tanda awal sebelum ada bahaya bagi pengguna jalan. Beberapa contoh dari rambu peringatan adalah perubahan kondisi jalan vertikal atau horizontal, kondisi jalan berbahaya, peraturan lalu lintas, dll. Beberapa figur dari rambu ini seperti pada Gambar 2.5.

Gambar 2.5 Rambu Peringatan

(Sumber : Kementerian Perhubungan, 2014)

2.2.2 Pengolahan Citra

Pengolahan citra adalah rekayasa untuk melakukan beberapa operasi pada gambar. Citra adalah gambar dua dimensi yang berasal dari gambar analog. Gambar analog dibagi menjadi kolom sebagai N dan baris sebagai M dan persilangan keduanya disebut dengan piksel. Operasi matematika yang dilakukan bertujuan untuk mendapatkan gambar yang disempurnakan dan mengambil beberapa informasi bermanfaat dari gambar. Terdapat dua tipe jenis pengolahan citra, yaitu analog dan pemrosesan citra digital. Pemrosesan citra digital adalah teknik untuk membantu manipulasi dari gambar digital menggunakan komputer maupun mini komputer seperti Raspberry Pi. Pemrosesan citra digital merupakan jenis pemrosesan sinyal *input* berupa gambar dan *output* dari sistem juga dalam bentuk gambar atau kerakteristik atau fitur yang terkait dengan gambar seperti pada Gambar 2.6 (Anbarjafari, 2014).

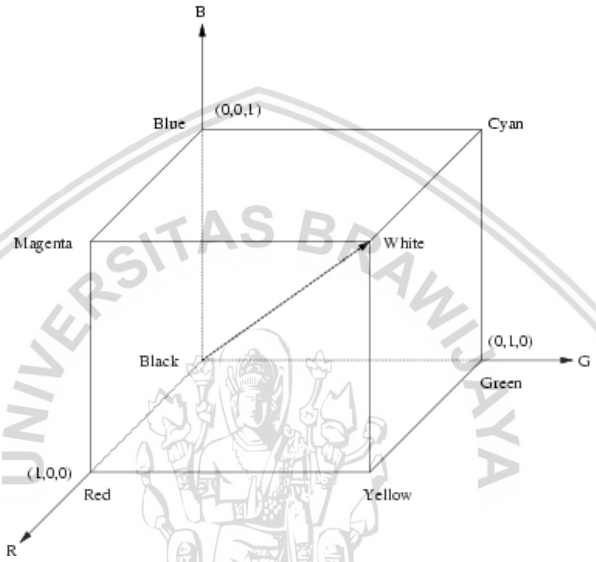
Gambar 2.6 Skematik Pengolahan Citra

(Sumber : Anbarjafari, 2014)

2.2.3 Konversi Citra RGB ke HSV

2.2.3.1 Citra BGR

Gambar model BGR atau RGB terdiri dari tiga komponen ruang warna utama yaitu biru, kuning, dan merah. Selain ketiga warna diatas, untuk menentukan warna tertentu adalah dengan menjumlahkan masing-masing komponen warna utama. Spektrum abu-abu, yaitu warna-warna primer dengan jumlah yang sama, spektrum ini terletak sepanjang garis yang menghubungkan *black* dan *white*. Untuk menentukan warna dalam RGB dapat melalui sistem koordinat *Cartesian* seperti pada Gambar 2.7 (OpenCV, 2018).



Gambar 2.7 Kubus Warna RGB

(Sumber : OpenCV, 2018)

Model aditif adalah warna baru yang terbentuk setelah percampuran cahaya berwarna seperti pada Gambar 2.8 sebelah kiri menunjukkan aditif pencampuran dari warna primer merah, hijau, dan biru untuk membentuk tiga warna sekunder yaitu kuning (merah+hijau), cyan (biru+hijau), magenta (merah+biru), dan putih (merah+hijau+biru).

Gambar 2.8 Model Aditif

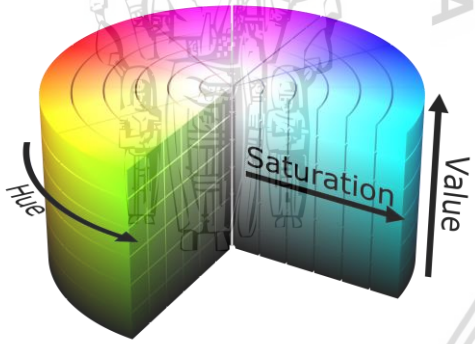
(Sumber : OpenCV, 2018)

Setiap warna dari RGB memiliki nilai minimum dan maksimum, nilai minimum dari tiap warna adalah nol (0) dan nilai maksimum 255 (setara dengan 8 bit). Warna RGB adalah warna yang dapat diterima dan dilihat oleh mata. Maka dari itu model RGB digunakan untuk monitor berwarna dan sebagian besar kamera video.

2.2.3.2 HSV *Color Space*

Alasan utama untuk menggunakan HSV karena HSV memisahkan *color information* (*chroma*) dari *intensity*/*lighting* (*luma*). Selain itu HSV merupakan salah satu warna yang dikenal oleh mata manusia. Ruang wana HSV berisi tiga deskriptor berbeda untuk mengklasifikasikan sebuah warna. Pertama adalah *hue* memiliki nilai dari 0-360°. Sehingga setiap warna contohnya saja merah akan berada di derajat yang sama walaupun dalam kondisi terang, gelap, dan tajam. Library OpenCV menyediakan *range* HSV dari 0-179 (OpenCV, 2018).

Saturasi untuk mewakili kemurnian (intens) warna atau berapa banyak warna yang dicampur dengan warna putih. Nilai batas dari saturasi yaitu 0-255. semakin tinggi saturasi maka warna akan terlihat semakin kaya sebaliknya warna akan terlihat pucat saat saturasi menurun dengan ketentuan nilai *value* tetap. Nilai *value* (kecerahan) menandakan banyaknya nilai warna hitam yang terkandung dalam suatu warna, maka dari itu seberapa pun nilai dari *hue* dan saturasi jika value menyentuh nilai 0 maka warna akan tetap gelap. Seperti Gambar 2.9, Berikut HSV *cylinder* yang mewakili nilai *hue*, saturasi, dan juga *value* (E. Petlenkov, 2015).



Gambar 2.9 HSV *Cylinder*

(Sumber : E. Petlenkov, 2015)

2.2.3.3 Konversi RGB ke HSV

Berikut langkah-langkah untuk menemukan batasan minimum dan maksimum dari masing-masing warna.

1. Pertama temukan nilai *R', B', G'* dari perbandingan masing-masing komponen dengan nilai maksimal gambar uint8 (gambar 8bit) seperti Persamaan 2.1, Persamaan 2.2, dan Persamaan 2.3.

$$R' = R/255 \quad (2.1)$$

$$G' = G/255 \quad (2.2)$$

$$B' = B/255 \quad (2.3)$$

2. Berikunya temukan nilai *cmax* pada Persamaan 2.4, *cmin* pada Persamaan 2.5, dan Δ pada Persamaan 2.6.

$$cmax = \max(R', G', B') \quad (2.4)$$

$$cmin = \max(R', G', B') \quad (2.5)$$

$$\Delta = cmax - cmin \quad (2.6)$$

3. Kalkulasi saturasi berbeda untuk setiap ketentuannya. Untuk nilai Δ sama dengan nol maka nilai *hue* otomatis 0°. Selain nilai Δ, nilai *cmax* juga mempengaruhi rumus untuk menemukan nilai *hue* seperti pada Persamaan 2.7.

$$H = \begin{cases} H = 60° \ x \ \left(\frac{G\prime - B\prime}{\Delta} + 0\right) & , cmax = R' \\ H = 60° \ x \ \left(\frac{B' - R\prime}{\Delta} + 2\right) & , cmax = G' \\ H = 60° \ x \ \left(\frac{R' - G\prime}{\Delta} + 4\right) & , cmax = B' \end{cases} \quad (2.7)$$

4. Kalkulasi saturasi pada variabel *S* juga tergantung nilai *cmax* seperti pada Persamaan 2.8.

$$S = \begin{cases} 0 \, , cmax = 0 \\ \frac{\Delta}{cmax} \, , cmax \neq 0 \end{cases} \quad (2.8)$$

5. Nilai *value* pada variabel *V* sama dengan nilai *cmax* Persamaan 2.9**.**

$$V = cmax \quad (2.9)$$

2.2.4 *Thresholding* (*inRange*)

Thresholding adalah salah metode segmentasi untuk memisahkan *foreground* (*region of interest*) dan *background*. Salah satu operasi *thresholding* pada *library* OpenCV disebut dengan fungsi *inRange()*. Fungsi ini akan memeriksa apakah elemen *array* piksel dari gambar saat ini terletak diantara elemen dari dua *array* lainnya (*lower* dan *upper*). Dimana nilai piksel yang berada dalam batas akan bernilai 1 sedangkan nilai 0 untuk nilai diluar batas. Batas-batas ini berasal dari nilai *hue* di Persamaan 2.10, *saturation* di Persamaan 2.11, dan *value* di Persamaan 2.12, dan *mask* pada Persamaan 2.13. Berikut cara kerja dari fungsi *inRange*() untuk menciptakan gambar biner.

$$dst(i)_{hue} = \Big(\big(lowerb \leq src(i)\big) \,\&\, \big(src(i) \leq upperb\big)\Big) \quad (2.10)$$

$$dst(i)_{saturation} = \Big(\big(lowerb \leq src(i)\big) \,\&\, \big(src(i) \leq upperb\big)\Big) \quad (2.11)$$

$$dst(i)_{value} = \Big(\big(lowerb \leq src(i)\big) \,\&\, \big(src(i) \leq upperb\big)\Big) \quad (2.12)$$

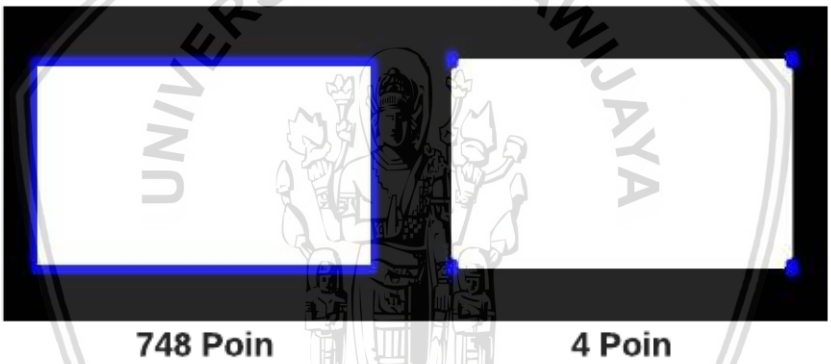
$$mask = dst(i)_{hue} \,\&\, dst(i)_{saturation} \,\&\, dst(i)_{value} \quad (2.13)$$

Variabel *dst(i)* adalah piksel destinasi ke *i* sebagai hasil dari persamaan tersebut (1 untuk putih dan 0 untuk hitam) di *frame* tujuan dimana gambar biner akan

disimpan. Sedangkan *src(i)* adalah piksel saat ini. *Lowerb* dan *upperb* adalah nilai skalar yang berisi *range* HSV (E. Petlenkov, 2015).

2.2.5 Kontur

Setelah melakukan *thresholding* gambar yang menghasilkan gambar biner, maka program mulai mencari gumpalan dari daerah piksel putih dan menyimpan dalam memori jika memenuhi syarat sebagai objek yang disebut dengan pencarian kontur. Untuk melakukan ini di pustaka OpenCV memiliki fungsi bernama *findcontours()*. Syarat menggunakan *findcontour*, gambar terlebih dahulu harus diubah menjadi 8bit (0-255) *single-channel* (HSV atau RGB memiliki 3-*channel*). Program pencarian untuk kontur, yang pada dasarnya vektor titik-titik yang membentuk titik-titik yang mengelilingi satu area warna yang sama. Benda-benda melengkung (misalnya lingkaran) vektor ini akan berisi sejumlah besar titik dan objek *rectangle*, hanya empat titik yang diperlukan (satu untuk setiap sudut), karena garis lurus dapat ditarik dari satu ke yang lain sehingga tidak ada titik terbentuk, akhirnya membentuk persegi panjang seperti pada Gambar 2.10 (OpenCV, 2018).



Gambar 2.10 Perbandingan Jumlah Titik Pada Kontur Persegi Panjang

(Sumber : OpenCV dev, 2018)

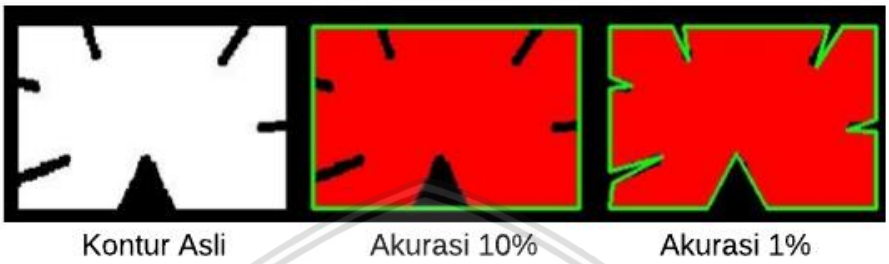
Setelah mengumpulkan informasi tentang semua gumpalan pada gambar. Program kemudian dapat menghitung luas kontur (*contour area*) dan koordinat pusat untuk area dalam setiap kontur. Area kontur dianggap objek jika nilai area mereka melampaui nilai minimum *default* atau yang ditentukan pengguna dan tetap dibawah maksimum. Kontur adalah alat yang berguna untuk analisis bentuk pada deteksi objek. Untuk akurasi lebih baik gunakan gambar biner (gunakan *threshold* atau *canny edge*) (OpenCV dev, 2018).

2.2.6 *Shape Detection*

Shape detection merupakan fase untuk mengenali bentuk dari rambu lalu lintas. Rambu memiliki bentuk tertentu. Suatu bentuk dapat dikenali oleh titik puncak (*vertice*) dan panjang-lebar sisi. Pencarian ciri tersebut dapat dilakukan dengan fase deteksi bentuk. Fase ini gabungan dari *approximation* dan *aspect ratio* dari kontur yang ditemukan. Kontur *approximation* adalah pendekatan

bentuk kontur (poligonal) asli ke bentuk dengan jumlah puncak/simpul yang lebih sedikit.

Terkadang saat menemukan kontur persegi panjang misalnya, bentuk dari persegi tidak sempurna (buruk) seperti pada Gambar 2.11**.** Fungsi *cv2.approxPolyDP* membantu agar bentuknya dapat diperkirakan. Titik-titik yang dihasilkan tergantung akurasi pada parameter *epsilon*. Parameter ini adalah jarak maksimum antara kurva orisinil dengan pendekatan (*approximation*) itu sendiri.



Gambar 2.11 Perbandingan akurasi fungsi *approximation*

(Sumber : OpenCV, 2018)

Selain menemukan titik *vertices* pada kurva, diperlukan seleksi luas kontur dengan fungsi *cv2.contourArea* agar tidak seluruh kontur perlu dianalisis bentuknya. Bagian ketiga dari deteksi bentuk ini seleksi berdasarkan jumlah *vertice* yang berasal dari jumlah titik yang ditemukan oleh fungsi *approx*. Maka sisi-sisi yang terbentuk dapat dijadikan identifikasi ciri. Selanjutnya seleksi kontur dengan aspek rasio. Aspek rasio adalah perbandingan jumlah panjang dan lebar dari suatu gambar (kontur). Cara mencari nilai panjang dan lebar suatu kontur adalah menggunakan *bounding rectangle* yang mengelilingi kontur tersebut. sehingga panjang dan lebar kontur sama dengan *width* dan *height bounding rect*. Dengan begitu ditemukan kepastian bentuk dari kontur-kontur tersebut (Puri, 2018).

Persamaan 2.14 bertujuan mengidentifikasi kategori *square*. sedangkan mengenali bentuk bulat memerlukan nilai radius seperti pada Persamaan 2.15, nilai aspek rasio pada Persamaan 2.16. Persamaan 2.17 untuk menemukan perbandingan luas lingkaran dengan *bounding box* (OpenCV, 2018).

- *Square*

$$ar = \frac{width}{height} \tag{2.14}$$

- *Circle*

$$radius = \frac{width}{2} \tag{2.15}$$

$$Aspect\ ratio = 1 - \frac{width}{height} \tag{2.16}$$

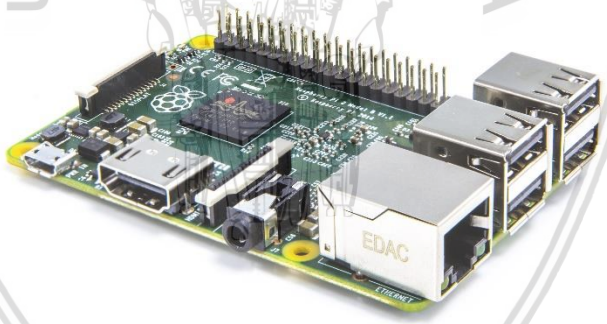
$$skala = 1 - \frac{luas\ kontur}{\pi radius^2} \tag{2.17}$$

2.2.7 Kebutuhan *Hardware*

2.2.7.1 Raspberry Pi

Raspberry Pi adalah komputer seukuran kartu kredit yang awalnya dirancang untuk penunjang pendidikan. Lain halnya dengan mikrokontroller, Raspberry Pi atau sering disebut dengan rasp-pi merupakan komputer berbasis *Linux* yang dapat berfungsi penuh. Distributor *Linux* saat ini menyediakan dua versi yang dioptimalkan untuk Raspberry Pi antara lain *Raspbian* dan *Pidora*. Walaupun Rasp-Pi bekerja lebih lambat dibandingkan dengan laptop modern atau *desktop* saat ini, tetapi Rasp-Pi tetap sebuah bentuk lengkap komputer dan mampu melakukan tugas sesuai yang diinginkan dan dengan konsumsi daya yang rendah.

Raspberry Pi memiliki kemampuan pendukung seperti CPU *board,* grafis, memori, dan USB *controller* seperti pada Gambar 2.12. Pada penelitian ini Raspberry Pi sebagai unit *processing* untuk mengimplementasikan proses pengolahan citra digital. Mengimplementasikan algoritma dari pengolahan citra digital tersebut maka membutuhkan OpenCV dan bahasa pemgrograman *Python*. Raspberry Pi memiliki spesifikasi kapasitas RAM 1 GB, slot yang dapat dipasangi pi *camera* maupun *WebCam*. Rasp-Pi dilengkapi prosesor ARM Cortex-A53 64-bit dengan kecepatan 1,2Ghz, hal ini juga menjadi alasan mengapa Rasp-Pi sebagai perangkat utama dari penelitian ini dan bukan mikrokontroller lainnya (opensource.com, n.d.).



Gambar 2.12 Raspberry Pi

(Sumber : raspberrypi.org*)*

2.2.7.2 *Speaker*

Pengeras suara atau *speaker* adalah alat untuk mengubah sinyal elektrik menjadi frekuensi suara atau *audio*. Cara kerja *speaker* yaitu dengan menggetarkan membran untuk menggetarkan udara hingga membentuk gelombang suara yang dapat didengarkan oleh telinga manusia. Salah satu contoh *speaker* yang digunakan seperti pada Gambar 2.13 (Logitech, n.d.).

Gambar 2.13 *Speaker*

(Sumber : logitech.com)

2.2.8 OpenCV (*Open Source Computer Vision*)

Opencv adalah *library* pendukung *computer vision* dan *machine learning* yang bersifat *open source*. *Library* ini terdiri dari 2500 algoritma. Algoritma ini digunakan untuk mendeteksi dan mengenali wajah, mengidentifikasi objek, mengklasifikasi tindakan manusia dari video, *tracking* pergerakan kamera, melacak objek yang bergerak, mengikuti gerakan mata, mencocokkan gambar. OpenCV dirilis dengan lisensi BSD dan oleh karena itu gratis untuk penggunaan akademis dan komersial. OpenCV dapat bekerja menggunakan bahasa pemrograman C++, C, *Python* dan *Java* dan mendukung sistem operasi seperti *Windows*, *Linux*, Mac OS, iOS dan *Android*. OpenCV dirancang untuk efisiensi komputasi dan fokus pada aplikasi *real-time*. Simbol dari library ini tertera pada Gambar 2.14 (Opencv dev, 2014).

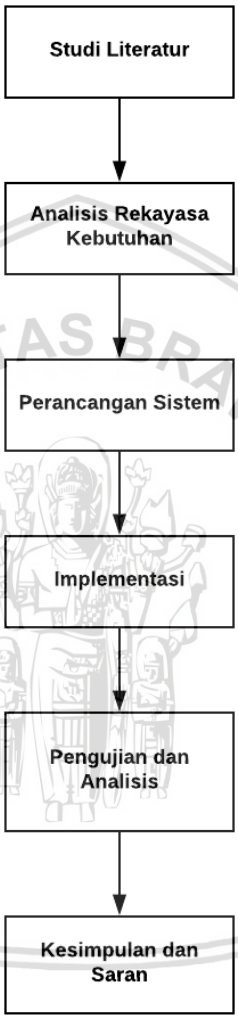
Gambar 2.14 OpenCV

(Sumber : opencv.org)

BAB 3 METODOLOGI

3.1 Alur Metode Penelitian

Tahapan pada penelitian ini yaitu studi literatur, rekayasa kebutuhan, perancangan sistem, implementasi, pengujian dan analisis, kesimpulan dan saran. Tahap-tahap dari penelitian ini tertera seperti Gambar 3.1.



Gambar 3.1 Tahap-tahap Metode Penelitian

3.2 Studi Literatur

Studi literatur berisi tentang referensi dari penelitian-penelitian yang sudah dilakukan sebelumnya dan dasar teori yang mendukung. Teori-teori yang dijadikan bahan pendukung penelitian ini antara lain:

1. Rambu Lalu Lintas
2. Pengolahan Citra Digital
3. Konversi Citra BGR ke HSV
4. *Thresholding (inRange)*

5. Kontur
6. *Shape Detection*
7. Kebutuhan *Hardware*
8. OpenCV

3.3 Analisis Kebutuhan

Kegiatan analisis kebutuhan yaitu menentukan kebutuhan yang di perlukan untuk melengkapi penelitian sistem deteksi dan pengenalan jenis berbasis Raspberry Pi. Berikut kebutuhan yang diperlukan dalam penelitan ini, yaitu :

Kebutuhan sistem dibagi menjadi kebutuhan perangkat keras terdiri dari beberapa perangkat keras untuk membangun penelitian ini. Dan Kebutuhan perangkat lunak adalah kebutuhan melingkupi program, aplikasi, atau *library* sebagai pendukung penelitian.

Berikutnya kebutuhan fungsional, disini akan dibahas tentang cara kerja sistem terhadap *input* yang masuk. Raspberry dengan program yang sudah tertanam dapat mengenali jenis rambu lalu lintas. Jika identifikasi gambar sudah selesai maka pengeras suara akan mengeluarkan nama jenis rambu. Terakhir kebutuhan non fungsional, Kebutuhan ini akan membentuk sistem yang optimal jika terpenuhi. Contoh kebutuhan non fungsional adalah batasan perancangan dan implementasi, karakteristik pengguna, lingkungan operasi, asumsi dan ketergantungan.

3.4 Perancangan Sistem

Konten dari bab ini berisi rancang bangun sistem dengan tujuan rumusan masalah yang telah dipaparkan sebelumnya dapat terselesaikan. perancangan dalam penelitian ini terbagi kedalam dua tahap, yaitu perancangan perangkat keras dan perancangan algoritma deteksi rambu (perangkat lunak).

3.5 Implementasi Sistem

Bab implementasi melaksanakan seluruh rancangan dari bab perancangan sistem, implementasi juga harus dilakukan sesuai tahap yang telah disusun. Pertama adalah perancangan perangkat keras. Rancangan ini dilakukan untuk membangun perangkat yang akan ditanam program deteksi rambu. Perangkat ini terdiri dari Raspberry Pi sebagai mesin pengolah bagi sistem. *Speaker* mengeluarkan hasil dari pengolahan sistem berupa peringatan kepada pengemudi.

Bagian terakhir yaitu algoritma deteksi rambu, algoritma ini sebagai unit proses dari sistem. Unit yang menerapkan sistem pengolahan citra untuk mendapatkan tujuan dari penelitian ini, yaitu mendeteksi rambu lalu lintas. Program ini terdiri dari fase segmentasi warna, identifikasi bentuk dan klasifikasi rambu untuk menggolongkan bentuk melalui aturan rambu lalu lintas di Indonesia.

3.6 Pengujian dan Analisis

Performa dan optimasi dari sistem perlu dilakukan untuk memastikan bahwa seluruh tujuan dari penelitian terlaksana, maka dari itu perlu melakukan pengujian dan analisis sistem yang dibangun. Pengujian dilakukan dengan skenario seperti waktu proses dan akurasi pengenalan warna, pengenalan bentuk, logika klasifikasi rambu. Jika pengujian masih belum berhasil maka kembali ke tahap analisis kebutuhan.

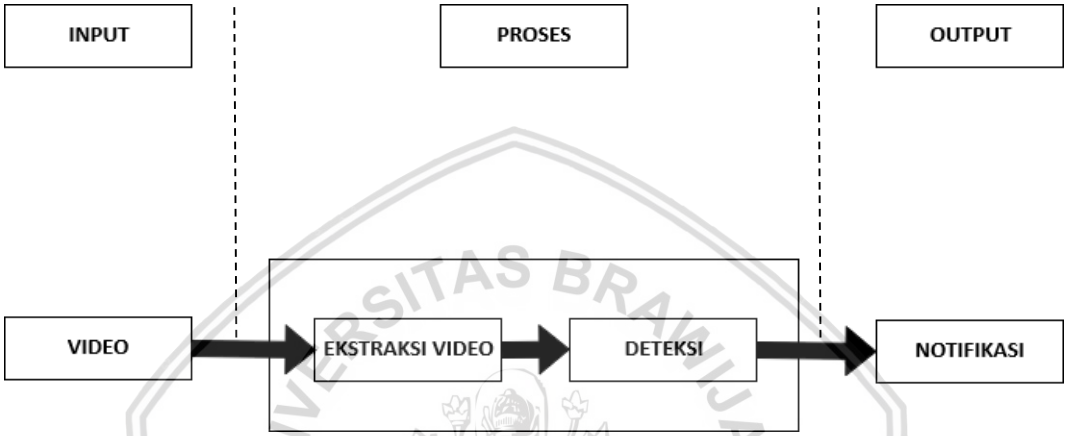
3.7 Kesimpulan dan Saran

Kesimpulan diperoleh setelah fase perancangan, implementasi dan pengujian hingga tahap analisis dari sistem. Kesimpulan didasarkan dari hasil pengujian dan analisis yang telah dilaksanakan. konten dari kesimpulan ini diharapkan dapat menjadi rujukan untuk penelitian selanjutnya dan membantu pengembangan teknologi *Driver Assistance System*. Didalam penulisan akhir terdapat saran yang bertujuan memberikan kemudahan kepada peneliti selanjutnya, apabila akan menerusan penelitian ini.

BAB 4 ANALISIS KEBUTUHAN

4.1 Gambaran Umum Sistem

Penelitian ini bertujuan untuk membangun sebuah sistem untuk mendukung teknologi pengenalan rambu jalan yang terdapat dalam *driver assistance system*. Sistem yang dibentuk adalah sistem deteksi dan pengenalan dari klasifikasi rambu dengan keluaran berupa suara bagi pengemudi. Berikut adalah diagram blok sistem seperti pada Gambar 4.1 dibawah ini.



Gambar 4.1 Blok Diagram Sistem

Blok diagram diatas memaparkan bentuk umum dari sistem, ada tiga bagian utama dari sistem ini, yaitu blok *input*, proses, dan *output*. video dari kamera atau *webcam* disebut sebagai blok *input*. Pengambilan video dilakukan *real-time*. Berikutnya blok proses, blok ini dibagi kedalam dua bagian, yaitu ekstraksi video dan deteksi. Kedua sub ini merupakan implementasi dari algoritma deteksi pada rambu lalu lintas. Ekstraksi *input* video merupakan salah satu implementasi *image processing*. Fase deteksi merupakan tahap untuk mencari daerah region dari rambu melalui hasil segmentasi gambar menggunakan *shape detection*. Dan yang terakhir notifikasi suara sebagai *output* bagi pengemudi.

4.2 Kebutuhan Fungsional

Sistem dapat dianggap berhasil ketika sudah memenuhi tujuan dari penelitian, dan berikut beberapa kebutuhan fungsional sistem.

1. Sistem dapat melakukan pembacaan dan ektraksi video yang diperoleh dari pengambilan video oleh kamera dilingkungan sekitar pengemudi.
2. Sistem dapat mendeteksi jenis, warna, dan bentuk rambu dari *frame* hasil ekstraksi video.
3. Sistem dapat mengklasifikasikan bentuk dari hasil fase deteksi sesuai dengan aturan pengenalan bentuk dan logika aturan rambu lalu lintas.
4. Sistem dapat menghitung jumlah waktu yang dibutuhkan untuk mendeteksi rambu.

5. Sistem dapat memberikan *ouput* suara melalui pengeras suara sesuai dengan keluaran program tersebut.

4.3 Kebutuhan Non-Fungsional

4.3.1 Batasan Perancangan Dan Implementasi

Batasan masalah digunakan untuk mengoptimasi rekayasa dan kerja sistem. Berikut batasan-batasan yang diberikan.

1. Sistem mulai bekerja setelah proses *booting* dari sistem operasi.
2. Sistem ini menggunakan *library* OpenCV sehingga mendukung pembacaan video secara *real-time*.
3. Posisi kamera saat mengambil video tegak lurus dengan rambu jalan.
4. Sistem dapat bekerja saat pagi, siang, maupun sore hari dengan cuaca cerah.
5. Rambu jalan dalam kondisi baik, tidak cacat dan rambu yang sesuai dengan peraturan Kementerian Perhubungan Republik Indonesia.
6. Pengambilan video dilakukan kendaraan dan bergerak maju.
7. *Output* utama sistem adalah notifikasi suara dari *speaker*.

4.3.2 Karakteristik Pengguna

Karakteristik pengguna diperuntukkan kepada peneliti selanjutnya untuk pengembangan sistem pengenalan citra rambu. Dengan adanya penelitian ini diharapkan mengoptimalkan pengembangan teknologi *Driver Assistance System*.

4.3.3 Lingkungan Operasi

1. Pengembilan video saat kondisi cuaca cerah pada pagi, siang, atau sore hari.
2. Posisi kamera tegak lurus dan tidak terhalang benda berwarna pekat sehingga tidak mengganggu pengambilan video.

4.3.4 Asumsi dan Ketergantungan

1. *Speaker* hanya akan mengeluarkan suara jika logika *ouput* sesuai dengan klasifikasi dari warna dan bentuk rambu lalu lintas.
2. Sistem hanya dapat bekerja jika sesuai dengan versi sistem operasi Raspberry, versi *library* OpenCV dan tipe bahasa pemrograman *Python* yang telah digunakan peneliti.

4.4 Kebutuhan Sistem

Kebutuhan ini bertujuan mendukung implementasi sistem dan membantu pengguna agar dapat melakukan monitoring terhadap sistem.

4.4.1 Kebutuhan Perangkat Keras

1. Raspberry Pi
   Raspberry pi atau Rasp-Pi adalah komputer dalam bentuk mini. Komputer ini memiliki sistem operasi yang menjalankannya dan seluruhnya disimpan ke dalam memori eksternal. Komputer ini sangat portabel dan mudah digunakan.

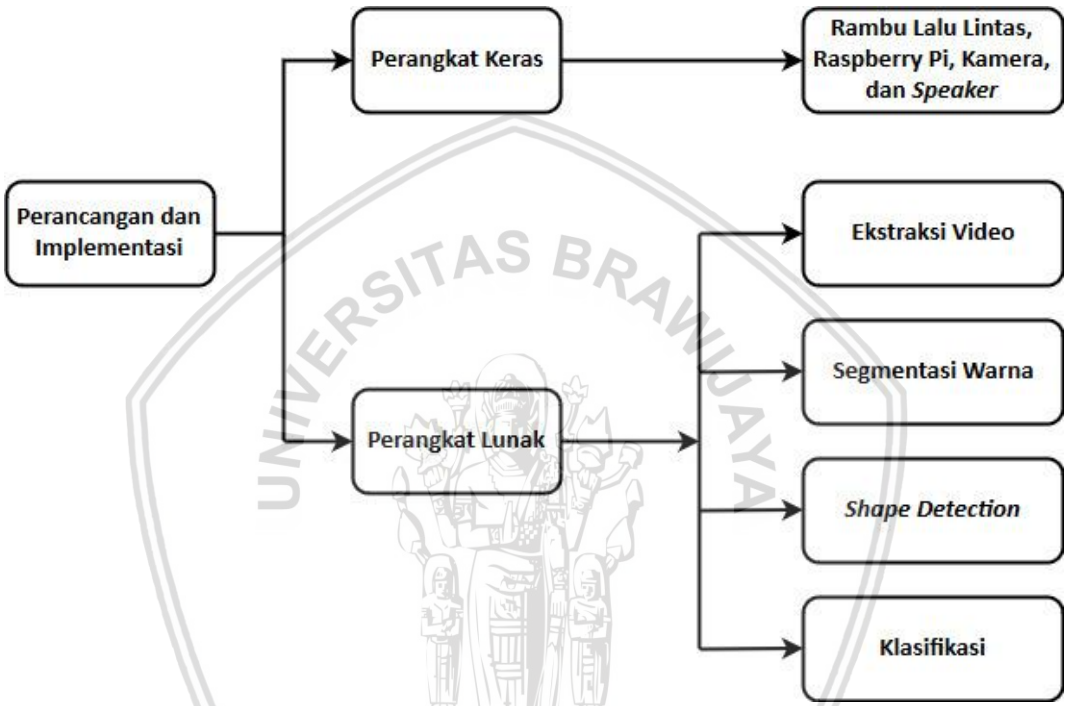
2. Kamera atau *Webcam*
   Perangkat ini digunakan untuk mengambil video sebagai *input* untuk diolah di blok proses sistem. Sistem yang dibangun menggunakan *Logitech webcam* 1280x720 piksel.
3. *Speaker*
   Speaker berguna sebagai keluaran sistem untuk mengeluarkan suara sesuai dengan *output* sistem.

4.4.2 Kebutuhan Perangkat Lunak

1. *Library* OpenCV
   Library ini sebagai penyokong implementasi pengolahan citra rambu lalu lintas. OpenCV mendukung operasi matematika dan mendukung pembacaan video secara *real-time*. Tipe dari OpenCV yang digunakan adalah OpenCV 3.3.0.
2. *Raspbian Stretch*
   Raspbian Stretch adalah satu versi sistem operasi ringan yang digunakan Raspberry Pi. *Raspbian* merupakan pengembangan dari sistem operasi *Linux*.
3. *Python*
   Python adalah salah satu bahasa pemrograman yang digunakan untuk membangun sistem ini. Versi *python* yang dipakai disini adalah *python* versi 3.
4. Modul *espeak*
   Modul espeak adalah modul text to speech dimana teks jenis rambu akan dikeluarkan kedalam bentuk suara di *speaker*.

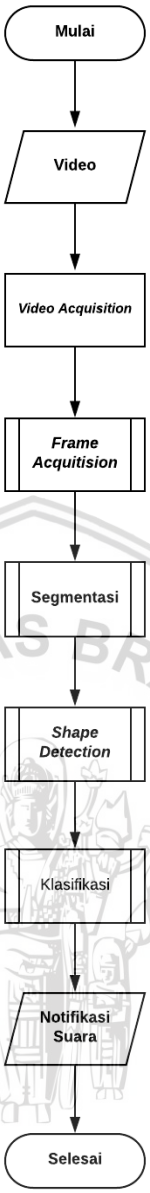
BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab ini menguraikan tahap perancangan dan implementasi sistem deteksi rambu menggunakan Raspberry Pi. Tahap-tahap tersebut dibagi menjadi dua bagian yaitu perangkat keras dan lunak. Perancangan *hardware* tersusun atas empat komponen, yaitu Raspberry pi, kamera, *speaker*, dan rambu. Sedangkan algoritma dari sistem deteksi rambu (*software*) akan melewati empat fase yaitu ekstraksi video, segmentasi menggunakan warna, pengenalan bentuk rambu, dan klasifikasi rambu jalan seperti pada Gambar 5.1 skema tahap tersebut dibentuk.



Gambar 5.1 Skema Perancangan dan Implementasi

Selain menyusun skema perancangan dan implementasi sistem, pada Gambar 5.2. akan diuraikan *flowchart* sistem deteksi rambu. Pertama video yang direkam oleh kamera diekstrak menjadi banyak *frame*, kemudian diolah per gambar dengan algoritma deteksi. Program ini berguna menemukan warna rambu yang diinginkan dengan segmentasi pada tiap warna secara bergantian pada gambar RGB. Hasil dari segmentasi gambar ini adalah gambar biner dari tiap warna. Setelah mendapatkan gambar biner maka untuk mendeteksi bentuk dari piksel harus dilakukan pencarian kontur dengan fungsi kontur. Kemudian tiap kontur ini diseleksi bentuknya berdasarkan operasi matematika. Setelah menemukan warna dan bentuk tiap kontur maka selanjutnya tahap klasifikasi. kategorisasi rambu dilakukan standar yang terdiri dari tiga kategori antara lain: rambu perintah, rambu larangan, rambu peringatan. Jika *frame* tersebut terdeteksi adanya rambu maka muncul notifikasi untuk pengemudi dari pengeras suara, jika tidak maka akan dilanjutkan pendeteksian ke *frame* selanjutnya.

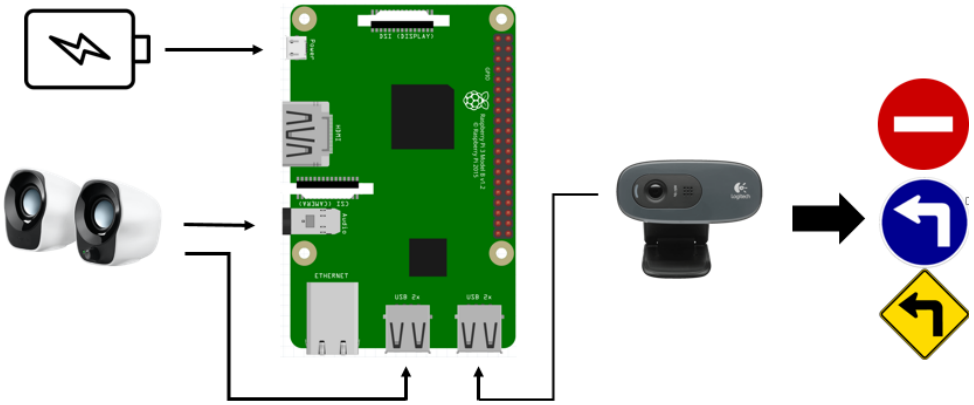


Gambar 5.2 *Flowchart* Sistem Deteksi Rambu

5.1 Perangkat Keras

5.1.1 Perancangan Perangkat Keras

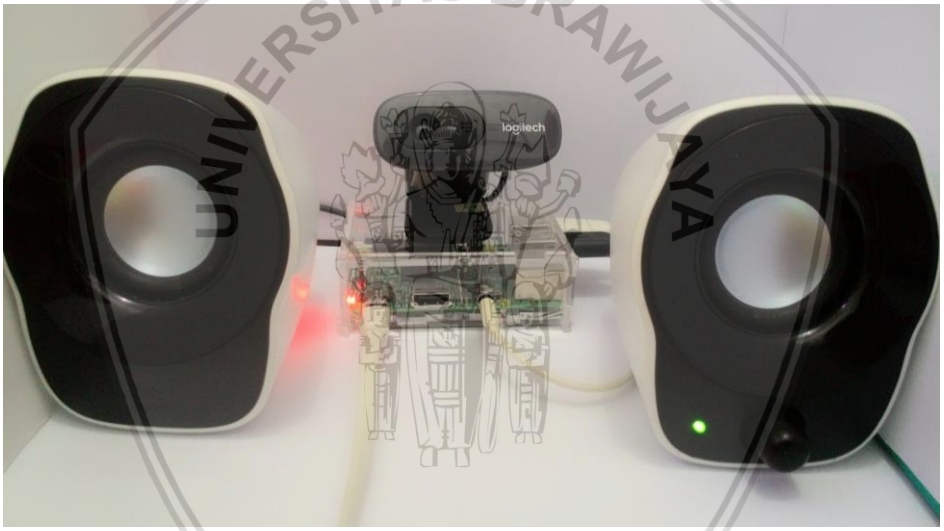
Pada tahap perancangan ini, kamera dan *speaker* sudah di inisialisasi didalam program dan siap digunakan. *Speaker* dan *webcam* tersambung ke port USB 2.0 menggunakan kabel USB. Kabel *audio speaker* tersambung dengan port video/*audio jack* sepeti pada Gambar 5.3. Mini komputer saat program berjalan kamera akan langsung mulai membaca dari lingkungan, lalu mencari keberadaan rambu dari video kondisi jalan dan setiap keluaran yang sesuai akan di notifikasi oleh *speaker*. Sistem ini akan bekerja secara *real-time* sehingga *output* dapat diprediksi waktunya.



Gambar 5.3 Perancangan Perangkat Keras

5.1.2 Implementasi Perangkat Keras

Seperti Gambar 5.4 berikut implementasi dari hasil perancangan komunikasi sistem tersebut. Setiap *device* sudah terhubung ke masing-masing port.



Gambar 5.4 Foto Alat

Mengakses kamera dengan Raspberry Pi melalui *library* OpenCV dapat dilakukan dengan *VideoCapture* seperti pada kode program di Tabel 5.1. Merekam video menggunakan kamera perlu untuk membuat objek *VideoCapture*. Argumen dari objek ini dapat berupa indeks perangkat (0) atau (-1) atau dapat dilakukan dengan nama file video “video.wmv”. bagian akhir program untuk tidak lupa merilis *capture* dengan *camera.release()*.

Tabel 5.1 kode program akses kamera

```
1  if not (False):
2      camera = cv2.VideoCapture(0)
3  else:
4      print("done")
5      camera = cv2.VideoCapture("video.wmv")
```

Ketika *speaker* ingin digunakan maka dilakukan *import* modul os pada kode program. Mengakses modul suara espeak tersebut dapat dilakukan seperti pada Tabel 5.2

Tabel 5.2 kode program akses kamera

```
1  os.system('espeak -vid+m1-s10 "Rambu Perintah"
2  2>/dev/null')
```

Berikut penjelesan dari baris program diatas :

- espeak, adalah salah satu modul suara yang dapat digunakan untuk memberi *output* suara
- -vid, merupakan potongan program untuk menyatakan bahwa bahasa yang digunakan adalah bahasa indonesia
- +m1, menjelaskan tipe suara yang digunakan yaitu *male* (laki-laki) jenis 1.
- -s10, untuk menentukan lama suara yaitu 10 ms
  "Rambu Perintah", adalah teks yang diubah ucapkan oleh pengisi suara

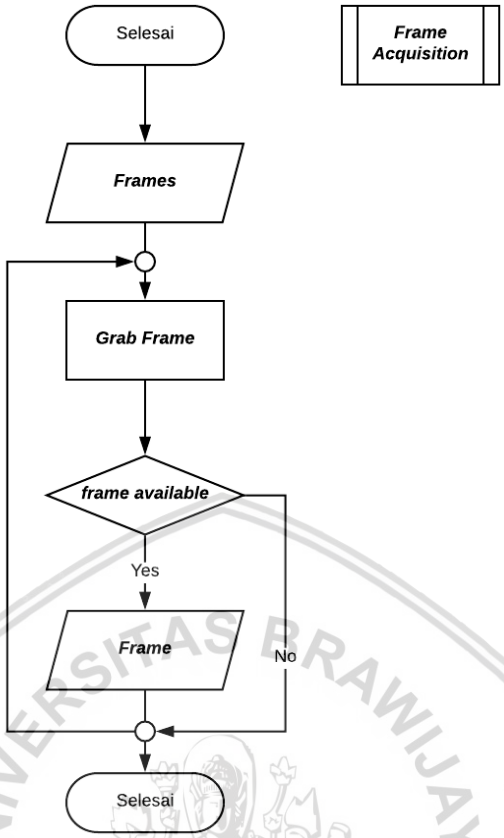
5.2 Perangkat Lunak

5.2.1 Perancangan Perangkat Lunak

Input dari proses deteksi ini adalah video hasil *capture* oleh kamera yang terpasang dengan Raspberry. Pengambilan video ini menggunakan library OpenCV berbasis *python*. Segmentasi warna, pengenalan bentuk dan klasifikasi adalah yang harus dilalui oleh video.

5.2.1.1 Ekstraksi Video

Video adalah sekumpulan gambar sekuensial dalam rentang waktu tertentu dengan tujuan membentuk gambar bergerak. Mengakses masing-masing *frame* ini disebut dengan *frame acquisition*. Seperti Gambar 5.5**,** berikut diagram dari *frame acquisition*.

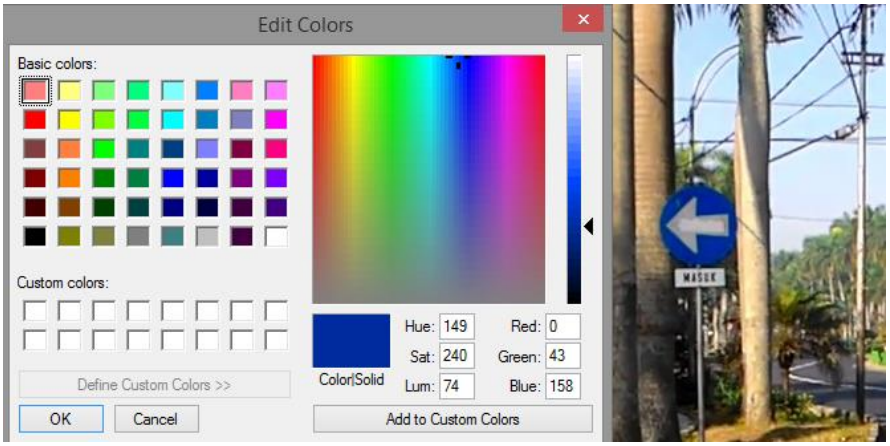


Gambar 5.5 *flowchart frame acquisition*

5.2.1.2 Segmentasi Warna

Segmentasi adalah proses untuk mendapatkan kemungkinan daerah piksel objek rambu yang disebut dengan *Region of Interest* (ROIs). Segmentasi berdasarkan warna artinya untuk mendapatkan regional piksel digunakan filter warna pada gambar HSV. Gambar HSV memiliki kelebihan yaitu memisahkan nilai dari *hue*, *saturation* (intens), dan kecerahan (*value*). Ruang warna HSV salah satu yang dapat dikenali oleh mata manusia. Namun tidak keseluruhan warna yang akan diambil, warna yang hanya perlu diambil adalah warna dari rambu lalu lintas (biru, merah, kuning) dan diberi *range*. nilai *lower* biru (100,50,50) dan *upper* biru (120,255,255). Nilai minimun *range* merah (170,50,50) dan maksimum (190,255,255). Sedangkan untuk nilai minimal kuning (20,100,100) dan maksimal (40,255,255). Berikut tahap-tahap untuk menemukan *range* dari *pixel* biru.

1. Temukan nilai R, G, B dari sebuah gambar rambu menggunakan salah aplikasi edit gambar seperti pada Gambar 5.6.



Gambar 5.6 Pencarian Nilai R, G, B

2. Cek nilai komponen R, G, dan B dari piksel warna biru kemudian cari nilai R', G', B' seperti pada Persamaan 2.1, Persamaan 2.2, dan pada Persamaan 2.3.

$$R' = 0/255$$
$$R' = 0$$
$$G' = 43/255$$
$$G' = 0.1686$$
$$B' = 158/255$$
$$B' = 0.6196$$

3. Setelah itu temukan nilai c maksimal dan c minimal seperti Persamaan 2.4 dan persamaan 2.5. kemudian selisih keduanya menggunakan Persamaan 2.6.

$$cmax = \mathrm{B}'$$
$$cmax = 0.6196$$
$$cmin = \mathrm{R}'$$
$$cmin = 0$$
$$\Delta = 0.6196 - 0$$
$$\Delta = 0.6196$$

4. Berbeda dengan *library* atau aplikasi lainnya, pada OpenCV nilai *hue* direpresentasikan dari 0-360°, agar format yang digunakan menjadi integer 8bit maka derajat tersebut dibagi menjadi dua bagian yaitu 0-179 dimana nilai 110 pada OpenCV sama dengan 220°. Setelah mendapat nilai *hue* seperti pada Persamaan 2.7 sebesar 223.67° maka di OpenCV nilai ini bisa dituliskan menjadi 111.83° atau 112°. Sekarang saatnya mengambil nilai range [H-10] dan [H+10] dari komponen ini sebagai batas atas dan bawah.

$$H = 60° \; x \; \left(\frac{-0.1686}{0.6196} + 4\right) \; , cmax = B'$$
$$H = 60° \; x \; 3.727 \; , cmax = B'$$

$H = 223{,}67°$, $cmax = B'$

5. Nilai saturasi didapat menggunakan Persamaan 2.8**.**

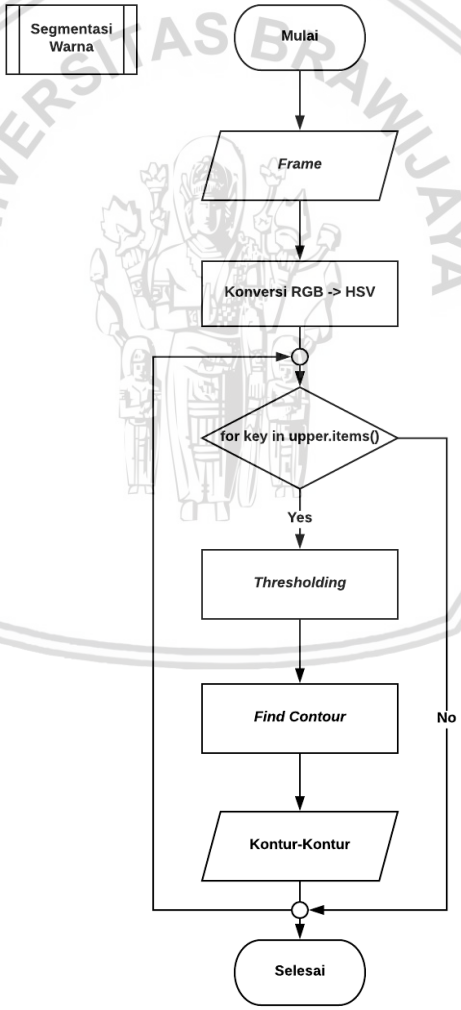
$$S = \frac{0.6196}{0.6196}$$

$$S = 1$$

6. Nilai *value* didapat menggunakan Persamaan 2.9**.**

$$V = 0.6196$$

Range HSV dari tiap warna tersebut yang menjadi ambang batas (*lower* dan *upper*) pada proses *thresholding*, dimana piksel yang masuk kedalam range akan bernilai 1 dan piksel lainnya dianggap 0. Kemudian saatnya untuk menemukan kontur dari piksel-piksel putih. Mengenali identitas warna dari gambar biner tersebut, maka proses pencarian kontur tidak dilakukan keseluruhan warna sekaligus, melainkan bergantian untuk tiap *range* seperti pada diagram alir pada Gambar 5.7.



Gambar 5.7 *Flowchart* Segmentasi Warna

5.2.1.3 *Shape Detection*

Setelah menemukan warna rambu, ada tiga jenis bentuk yang akan dideteksi yaitu bulat, *triangle* (segitiga), dan persegi (*diamond*). Menemukan diperlukan dua batasan yaitu *vertice* dan panjang-lebar. Misalnya ditemukan 4 puncak pada sebuah kontur, maka ada kemungkinan bahwa itu adalah persegi dimana adalah bagian dari bentuk rambu peringatan. selain persegi ada lingkaran yang memiliki titik lebih dari sama dengan 8, segitiga memiliki 3 titik, dan oktagonal dengan 8 titik seperti pada Tabel 5.3. Namun tidak cukup sampai disitu saja perlu memastikan panjang dan lebar dari kontur tersebut agar rambu lebih spesifik lagi disebut dengan persegi.

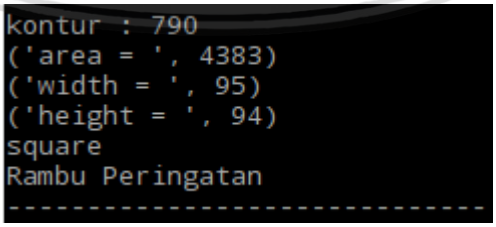
Tabel 5.3 Bentuk dan Jumlah *Vertice*

Bentuk	*Vertice* (Titik)
Circle	>=8
Square	4
Triangle	3
Oktagonal	8

Aspek rasio adalah salah satu cara menemukan perbandingan panjang-lebar suatu gambar. Dengan bantuan fungsi *bounding rectangle* yang mengelilingi kontur dengan luas minimum, maka akan didapatkan nilai panjang dan lebar kontur berdasarkan width-height dari *bounding rect* tersebut. Berikut langkah untuk mendapatkan batas aspek rasio dari kotak dan lingkaran.

- **Kotak (*diamond*)**

1. Aspek rasio adalah perbandingan panjang dan lebar suatu gambar, dalam kasus ini panjang dan lebar dari daun rambu peringatan (*diamond*) adalah nilai *width* dan *height* dari *bounding rectangle* seperti pada Gambar 5.8.



Gambar 5.8 nilai *area*, *width*, dan *height*

2. Nilai *width*(*w*) dan *height*(*h*) dari contoh rambu berbentuk kotak adalah 109, 107, maka nilai aspek rasio didapatkan berdasarkan persamaan 2.14.

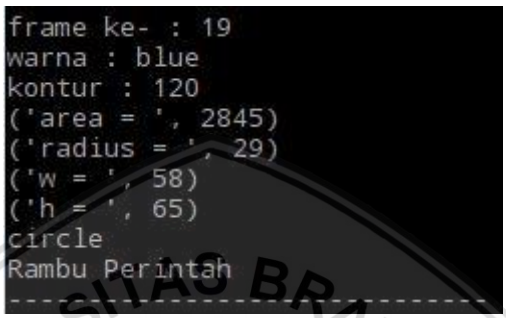
$$ar = \frac{95}{94}$$

$$ar = 1.010$$

3. Berdasarkan nilai *ar* yang didapatkan maka range dari aspek rasio untuk bentuk kotak adalah 0.95 hingga 1.1.

- **Lingkaran**

1. Berbeda dengan bentuk kotak, selain mencari aspek rasio dari lingkaran juga pebandingan luas kontur berdasarkan fungsi *contour area* dan luas kontur berdasarkan nilai radius. Sebelum menentukan nilai batas *aspect ratio* dan skala luas lingkaran maka lebih dahulu mencari nilai luas kontur, *radius*, *w* dan *h* seperti pada Gambar 5.9.



Gambar 5.9 nilai *area*, *radius*, *w*, dan *h*

2. Mencari nilai radius dari lingkaran berdasarkan Persamaan 2.15.

$$radius = 58/2$$

$$radius = 29$$

3. Mencari nilai perbandingan *w* dan *h* dari *bounding rectangle* sebagai representasi nilai *w* dan *h* lingkaran seperti pada persamaan 2.16.

$$aspect\ ratio = abs(1 - \left(\frac{58}{65}\right))$$

$$aspect\ ratio = abs(1 - 0.8923)$$

$$aspect\ ratio = 0.1076$$

4. Terakhir mencari nilai perbandingan luas kontur lingkaran berdasarkan perbandingan nilai luas kontur dan perhitungan luas lingkaran menggunakan radius seperti pada Persamaan 2.17.

$$skala = \text{abs}(1 - (\frac{2845}{3.14\ x\ 29^2})$$

$$skala = \text{abs}(1 - (\frac{2845}{2640.74})$$

$$skala = \text{abs}(1 - (\frac{2845}{2640.74})$$

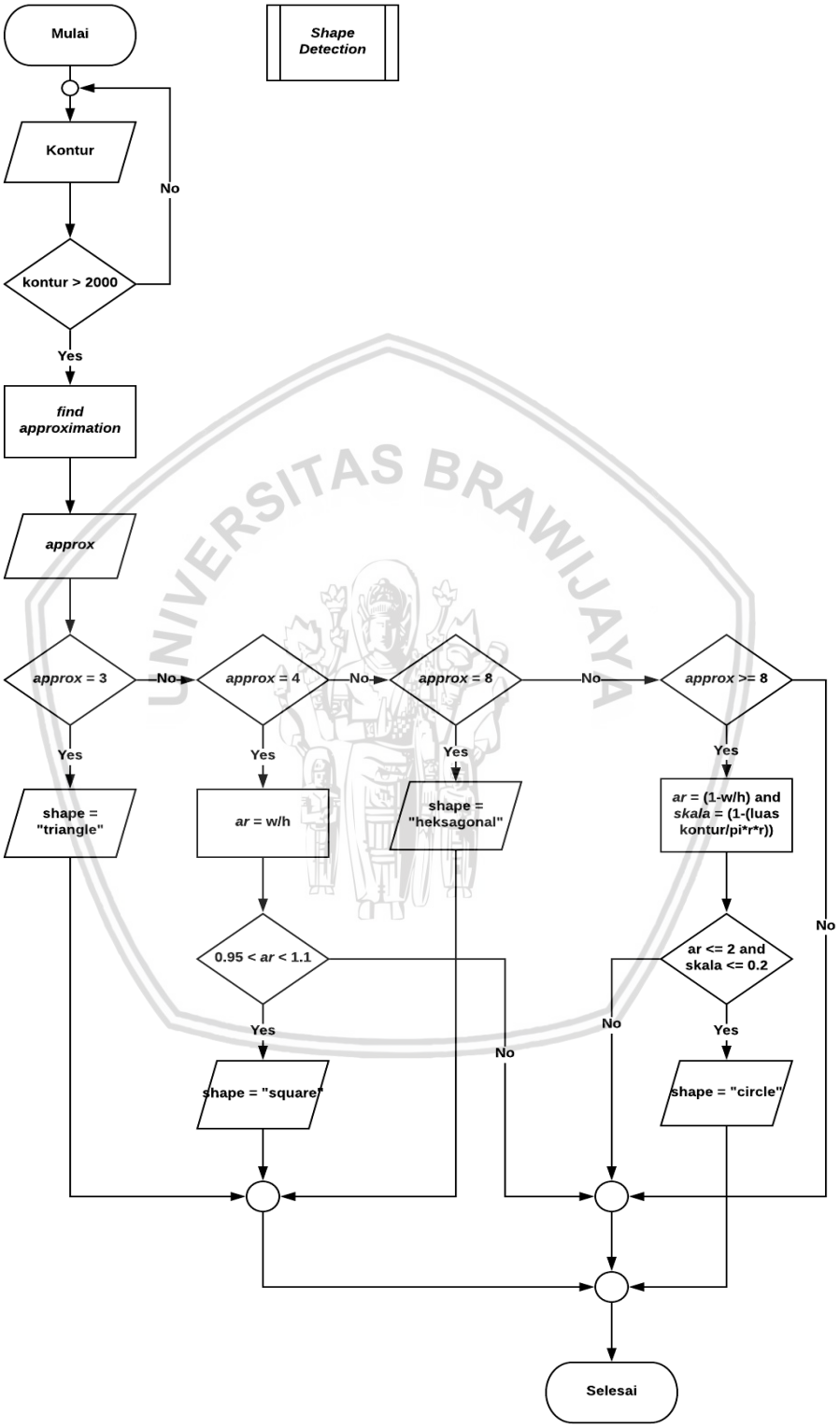
$$skala = \text{abs}(1 - 1.077)$$

$$skala = \text{abs}(-0.077)$$

$$skala = 0.077$$

5. Maka diperoleh nilai ambang batas aspek rasio dan skala luas lingkaran yaitu 2 dan 0.2.

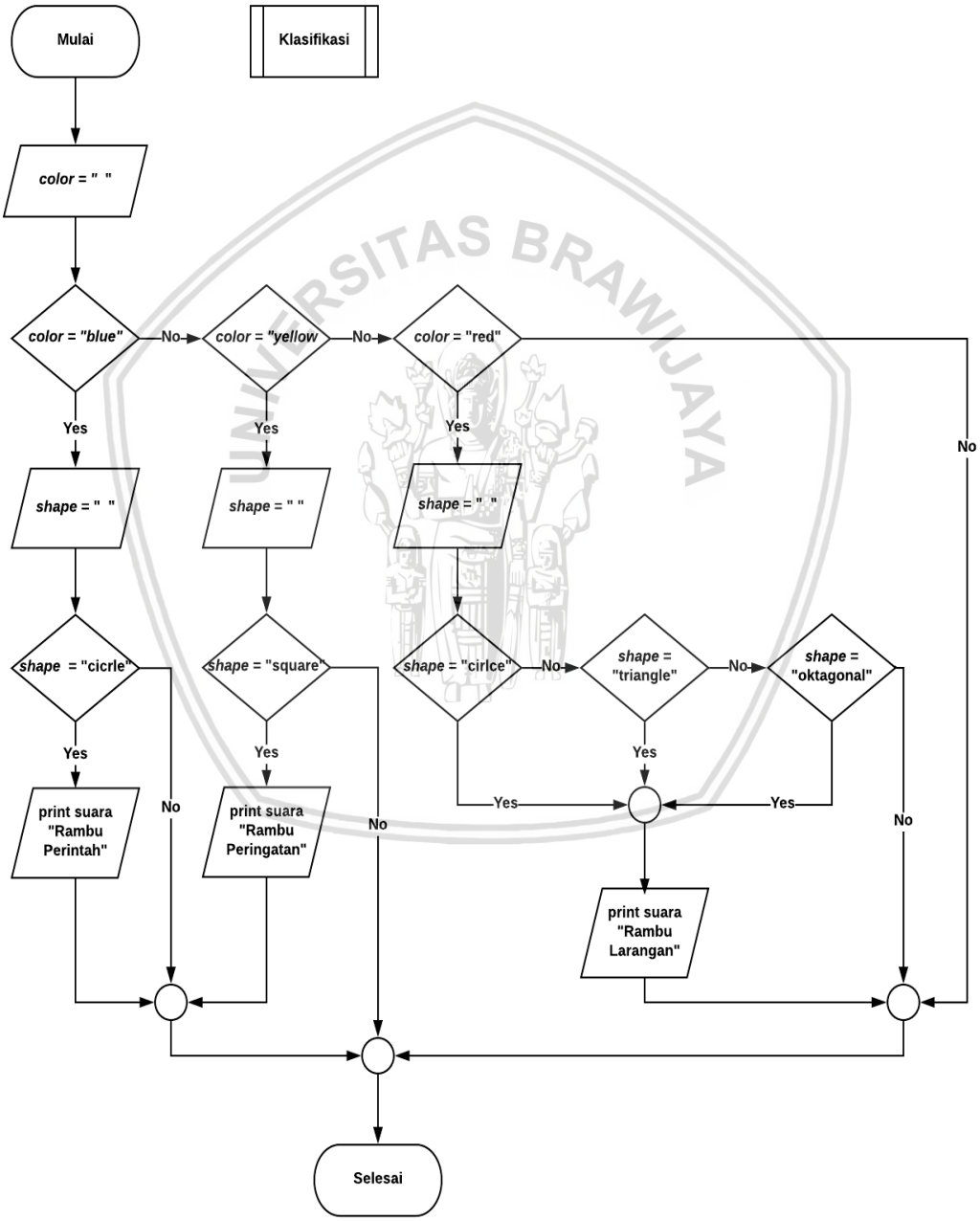
Setelah menemukan nilai *approx* dan batas dari aspek rasio dan luas area untuk masing-masing bentuk daun rambu lalu lintas (peringatan, perintah, dan larangan) maka dapat disusun seperti diagram alir di Gambar 5.10.



Gambar 5.10 Diagram Alir *Shape Detection*

5.2.1.4 Klasifikasi

Setiap rambu perintah, peringatan, dan larangan memiliki ciri khas tersendiri dan hal ini membantu rambu lebih informatif. Ciri setiap rambu ditandai oleh warna dan bentuk rambu. Setelah menemukan kedua identitas ini dari tahap sebelumnya, berdasarkan aturan dari Kementerian Perhubungan. Maka dapat ditemukan kemungkinan apakah kontur tersebut adalah representasi dari rambu atau bukan. Dengan begitu akan terdeteksi keberadaan rambu seperti *flowchart* pada Gambar 5.11. Dalam penelitian ini ada 3 jenis rambu yang akan dideteksi, yaitu rambu larangan, rambu perintah, dan rambu peringatan.



Gambar 5.11 *Flowchart* Klasifikasi Rambu

5.2.2 Implementasi Perangkat Lunak

5.2.2.1 Ekstraksi Video

Implementasi dari perancangan sebelumnya seperti pada Tabel 5.4. Hasil balikan dari *camera.read()* adalah *boolean* (*true/false*). Jika *frame* dibaca dengan benar maka akan memberikan nilai balik *True*. Jika tidak maka program selesai.

Tabel 5.4 Kode Program *Frame Acquisition*

```
1  while True:
2      (grabbed, frame) = camera.read()
```

5.2.2.2 Segmentasi Warna

Pertama-tama konversi RGB ke HSV terletak pada baris ke-18 untuk memudahkan proses binerisasi. Berdasarkan batas minimal dan maksimal nilai *hue, saturation,* dan *value* akan diseleksi warna dari rambu. Setiap warna diproses dengan perulangan. Terakhir pencarian kontur-kontur gambar biner dengan fungsi kontur seperti pada Tabel 5.5 baris ke-22.

Tabel 5.5 Kode Program Segmentasi Warna

```
1  from klasifikasi import klasifikasi_rambu
2  import numpy as np
3  import math
4  import argparse
5  import imutils
6  import cv2
7  import time

8  contours = {}
9  approx = []
10 scale = 2
11 starttime = 0.0
12 endtime = 0.0
13 waktu = 0.0
14 color = "undetected color"
15 starttime = time.clock()
16 lower =
17 ({'blue':(100,50,50),'yellow':(20,50,50),'red':(170,50
   ,50)})
18 upper =
   ({'blue':(120,255,255),'yellow':(40,255,255),'red':(19
   0,255,255)})
19 hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
20     for color, value in upper.items():
21         mask = cv2.inRange(hsv, lower[color],
   upper[color])
22         (contours, hierarchy) =
   cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHA
   IN_APPROX_SIMPLE)
23 dd = klasifikasi_rambu()
```

5.2.2.3 *Shape Detection*

Banyaknya benda berwarna sama dengan rambu di lingkungan menjadi alasan perlunya pengenalan bentuk dari objek tersebut. Implementasi fungsi untuk menemukan *vertice* dari sebuah kontur menggunakan fungsi kontur *approx* yang dijabarkan seperti pada baris ke-4. Kemudian seleksi luas kontur, dimana hanya kontur dengan luas melebihi 2000 atau ditemukannya titik pada kontur yang akan diseleksi. Membuktikan jenis bentuk dari kontur dilakukan dengan seleksi kembali berdasarkan jumlah titik dari proses *approximation* dan panjang-lebar kontur pada Tabel 5.6.

Tabel 5.6 Kode Program *Shape Detection*

```
1   for i in range(0,len(contours)):
2               shape="undetected"
3               a,b,c,d = cv2.boundingRect(contours[i])
4               approx =
    cv2.approxPolyDP(contours[i],cv2.arcLength(contours[i]
    ,True)*0.02,True)
5               if(abs(cv2.contourArea(approx))<2000 or
    not(cv2.isContourConvex(approx))):
6                   continue
7               if(len(approx)==3):
8                   shape = "triangle"
9                   print(shape)
10              if(len(approx)==4):
11                  area = cv2.contourArea(approx)
12                  x,y,w,h = cv2.boundingRect(approx)
13                  ar = w/float(h)
14                  print(int(area))
15                  shape = "square"
16                  if (ar >= 0.95 and ar<= 1.1):
18                      print(shape)
19              elif(len(approx)>=8):
20                  area = cv2.contourArea(approx)
21                  x,y,w,h = cv2.boundingRect(approx)
22                  print(int(area))
23                  if (ar == 1):
24                      shape = "oktagonal"
25                      print(shape)
26                  radius = w/2
27                  elif (abs(1-(float(w)/h))<=2 and abs(1-
28  (area/(math.pi*radius**2)))<=0.2):
                        shape = "circle"
29                      print(shape)
30              else:
31                  print(shape)
32              klas = dd.kelas(color,shape)
33              endtime = time.clock()
34              waktu = endtime - starttime
```

5.2.2.4 Klasifikasi

Tahap pertama klasifikasi adalah pengkondisian identitas warna dari kontur yang ditemukan, alasannya adalah ciri utama dari jenis-jenis rambu ini terletak pada warna. Contoh rambu warna kuning memiliki nilai kepentingan lebih dari pada rambu warna biru, maka dari itu disebut dengan rambu peringatan sebagai tanda bahaya di jalan. Mengkategorikan bentuk, setelah menemukan warna dan bentuk tertentu maka dapat disimpulkan jenis dari rambu tersebut. Kesimpulan ini ditentukan melalui *output* suara dengan modul *espeak* seperti pada baris program ke-12, baris ke-19, dan baris ke-30 di Tabel 5.7.

Tabel 5.7 Kode Program Klasifikasi Rambu

```
1  import cv2
2  import os,sys

3  class klasifikasi_rambu:
4       def __init__(self):
5            pass
6       def kelas(self, color, shape):
7            rambu = "NO SIGN"
8            if color in ['blue']:
9                 if shape in ['circle']:
10                     rambu = "Rambu Perintah"
11                     print(rambu)
12                     os.system('espeak -vid+m1-s100
   "Rambu Perintah" 2>/dev/null')
13                else:
14                     print(rambu + " " + color)
15           elif color in ['yellow']:
16                if shape in ['square']:
17                     rambu = "Rambu Peringatan"
18                     print(rambu)
19                     os.system('espeak -vid+m1-s100
   "Rambu Peringatan" 2>/dev/null')
20                else:
21                                print(rambu + " " +
   color)
22           elif color in ['red']:
23                          if shape in ['circle']:
24                                rambu = "Rambu
   Larangan"
25                                print(rambu)
26                                os.system('espeak -
   vid+m1-s100 "Rambu Larangan" 2>/dev/null')
27                          elif shape in ['triangle']:
28                          rambu = "Rambu Larangan"
29                                print(rambu)
30                                os.system('espeak -
   vid+m1-s100 "Rambu Larangan" 2>/dev/null')
31                elif shape in ['oktagonal']:
32                     rambu = "Rambu Larangan"
33                                print(rambu)
```

BAB 6 PENGUJIAN DAN ANALISIS

6.1 Pengujian Pembacaan Video

6.1.1 Tujuan Pengujian

Pengujian ini memiliki fungsi untuk menguji performa dari sistem saat pembacaan video dari kamera.

6.1.2 Pelaksanaan Pengujian

Pengujian dilakukan menggunakan *webcam logitech 720p* yang tersambung ke USB port 2.0 Raspberry Pi.

6.1.3 Prosedur Pengujian

Berikut prosedur yang dilakukan untuk pengujian ini antara lain:

1. Membuka aplikasi monitoring kerja Raspberry Pi yaitu VNC.
2. Menyiapkan program deteksi dan Raspberry Pi.
3. *Webcam* tersambung ke port USB 2.0 menggunakan kabel USB.
4. Menjalankan program "deteksi.py" pada terminal aplikasi VNC.
5. Melihat perubahan lampu led pada kemera. Jika lampu berwarna hijau berarti berfungsi dan dapat membaca video.

6.1.4 Hasil Pengujian

Berikut Gambar 6.1 saat led hijau *webcam* menyala artinya berhasil membaca kondisi jalan dan siap digunakan untuk merekam.

Gambar 6.1 *Webcam* menyala

Berdasarkan percobaan yang telah dilakukan sebanyak 10 kali maka didapat hasil seperti pada Tabel 6.1.

Tabel 6.1 Tabel Pengujian Pembacaan Video

Percobaan	Kondisi Led	Status
1	Berwarna hijau	Berhasil
2	Led Padam	Gagal
3	Led Padam	Gagal
4	Berwarna hijau	Berhasil
5	Berwarna hijau	Berhasil
6	Berwarna hijau	Berhasil
7	Led Padam	Gagal
8	Berwarna hijau	Berhasil
9	Berwarna hijau	Berhasil
10	Berwarna hijau	Berhasil

6.1.5 Analisis Hasil Pengujian

Percobaan pembacaan oleh *webcam* berhasil dilakukan karena kabel USB *webcam* sudah terpasang pada USB port 2.0 Raspberry Pi sehingga saat program berjalan *webcam* akan langsung bekerja.

6.2 Pengujian Parameter *Contour Area*

6.2.1 Tujuan Pengujian

Tujuan pengujian ini adalah menemukan nilai ambang batas yang tepat dari parameter *contour area* berdasarkan jarak rambu dengan sistem.

6.2.2 Pelaksanaan Pengujian

Pengujian nilai parameter dimulai dari jarak 1-7 meter. Kemudian mencari nilai ideal area kontur pada *range* jarak tersebut.

6.2.3 Prosedur Pengujian

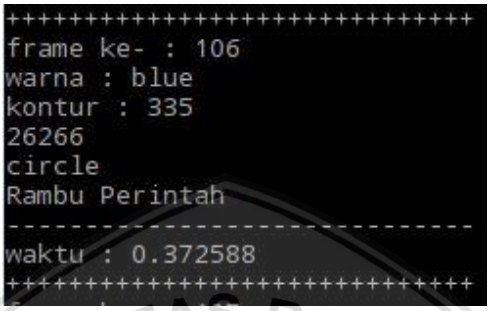
Pengujian ini melewati prosedur sebagai berikut.

1. Menyiapkan Raspberry Pi dan kamera tepat didepan rambu.
2. Melakukan pengambilan gambar rambu pada jarak 1 meter, 3 meter, 5 meter, dan 7 meter.
3. Mengamati nilai kontur rambu dari masing-masing jarak yang telah diukur pada terminal.
4. Menganalisis perubahan nilai kontur dari seluruh jarak untuk mendapatkan nilai ambang batas area kontur yang tepat.

6.2.4 Hasil Pengujian

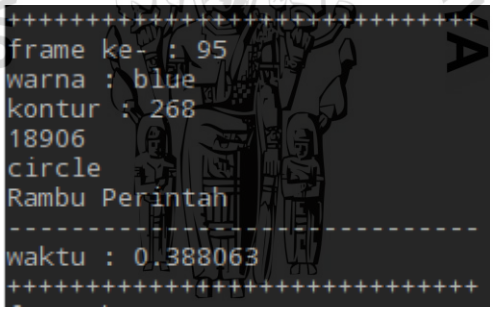
Setelah pengujian jarak maka diperoleh hasil dari masing-masing area kontur rambu saat terdeteksi sebagai berikut:

1. Jarak 1 m
   Pada pengambilan gambar pada jarak 1 meter seperti pada Gambar 6.2 nilai area kontur melebihi 26000 . rambu dapat terdeteksi warna yang benar namun identifikasi bentuk salah (*undetected*).



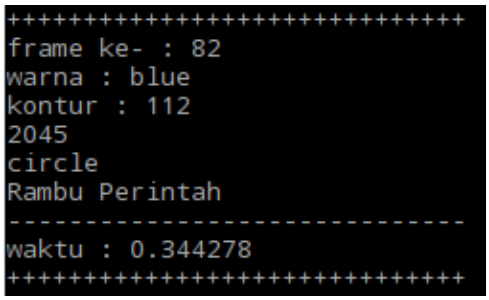
Gambar 6.2 Area Kontur Pada Jarak 1m

2. Jarak 3 m
   Rambu pada jarak 3 m dapat dikenali jenisnya dengan benar seperti pada Gambar 6.3. Nilai luas area dari kontur ini yaitu 18906.



Gambar 6.3 Area Kontur Pada Jarak 3 m

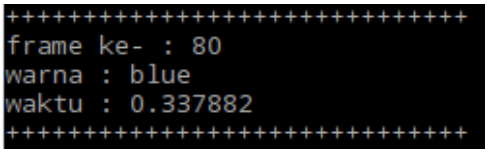
3. Jarak 5 m
   Pengambilan gambar dari jarak 5 m masih ideal untuk mengidentifikasi warna dan bentuk rambu tertera pada Gambar 6.4. Jarak ini area kontur rambu mencapai 2045.



Gambar 6.4 Area Kontur Pada Jarak 5 m

4. Jarak 7 m

   Pada jarak ini pendeteksian rambu sudah tidak ideal. Tidak terdeteksi warna dan bentuk rambu tersebut seperti di Gambar 6.5.



Gambar 6.5 Area Kontur Pada Jarak 7 m

6.2.5 Analisis Hasil Pengujian

Berdasarkan pengujian luas kontur berdasarkan jarak yang telah ditentukan maka didapatkan hasil seperti pada Tabel 6.2. Pendeteksian warna dan bentuk yang benar ditandai dengan simbol √ dan × simbol untuk melihat kesalahan deteksi. Berdasarkan pengujian ini, sistem dapat mengidentifikasi warna namun tidak dapat identifikasi bentuk pada jarak 1 m dengan sendirinya tidak dapat mendeteksi klasifikasi rambu. Pada jarak 7 m sistem tidak dapat mendeteksi rambu karena tidak ditemukannya warna dan bentuk sama sekali.

Tabel 6.2 Luas Kontur

Jarak (m)	Warna	Bentuk	Luas Kontur
1	√	×	>26000
3	√	√	18000
5	√	√	2000
7	×	×	<2000

6.3 Pengujian Ketepatan Deteksi Jenis Rambu

6.3.1 Tujuan Pengujian

Pengujian ini bertujuan untuk melihat performa sistem sesuai dengan perancangan sistem dan menghasilkan keluaran jenis rambu yang tepat.

6.3.2 Pelaksanaan Pengujian

Pengujian dilakukan dengan menjalankan program deteksi yang sudah tersimpan dalam Raspberry Pi. Keluaran dari program dapat dilihat pada terminal dan *frame* yang menampilkan kondisi jalan saat itu.

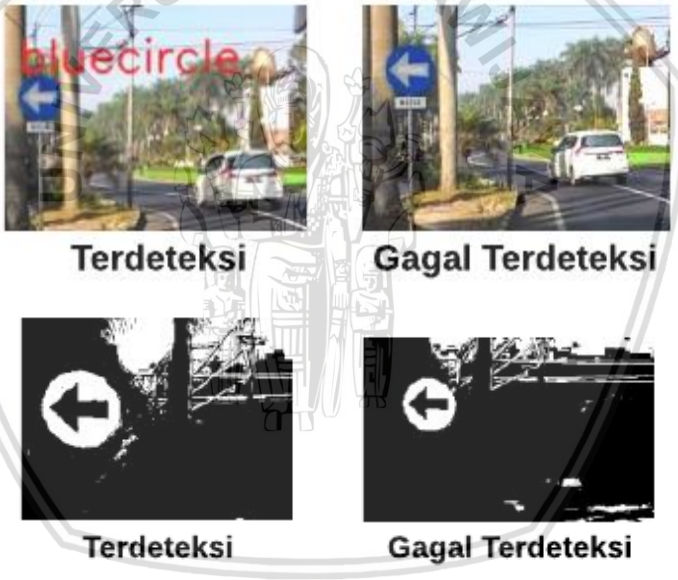
6.3.3 Prosedur Pengujian

1. Menghubungkan Raspberry dengan *power bank* (led merah menyala).
2. Webcam terpasang ke port 2.0 Raspberry Pi.
3. Menghubungkan mini komputer (Raspberry Pi) dengan PC menggunakan kabel lan (led hijau berkedip).

4. Monitoring kerja sistem melalui *frame* aplikasi monitoring Raspberry Pi.
5. Meletakkan perangkat *webcam* didepan kendaraan dan menghadap lurus ke jalan.
6. Menjalankan program “deteksi.py” pada terminal Rasp-Pi.
7. Menguji sistem saat kendaraan berjalan menuju rambu pada jarak 2-5 meter dengan kecepatan 30km/jam.
8. Mengamati dan mencocokan keluaran dari *speaker* dan *frame*.
9. Menghitung jumlah *frame* dari kesuluruhan total *frame* dari masing-masing parameter (warna, bentuk, jenis) dari seluruh rambu.

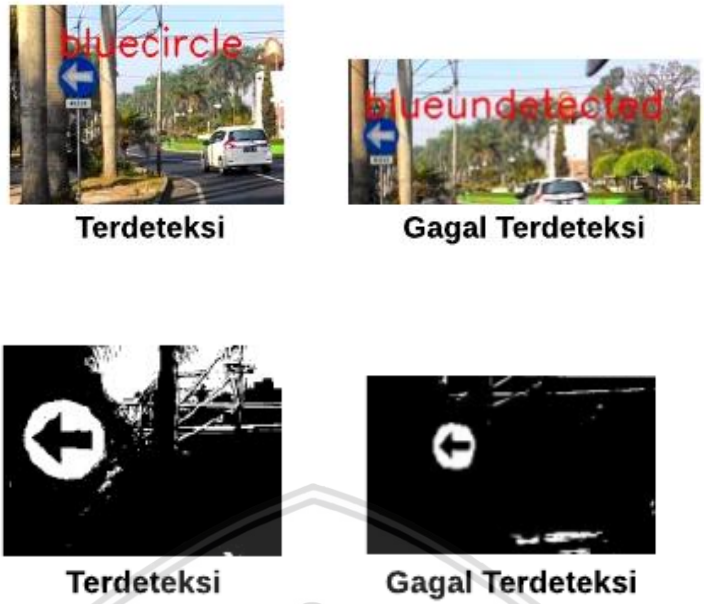
6.3.4 Hasil Pengujian

a. Terdeteksi Warna
   Rambu akan terdeteksi jika berada *range* HSV pada program seperti pada Gambar 6.6. Pengujian warna pada objek kontur sangat diperlukan karna warna merupakan salah satu syarat utama untuk membandingkan rambu dengan objek lainnya, jika tidak terpenuhi maka *frame* yang terlihat ada rambu tidak akan terdeteksi.



Gambar 6.6 Deteksi Warna

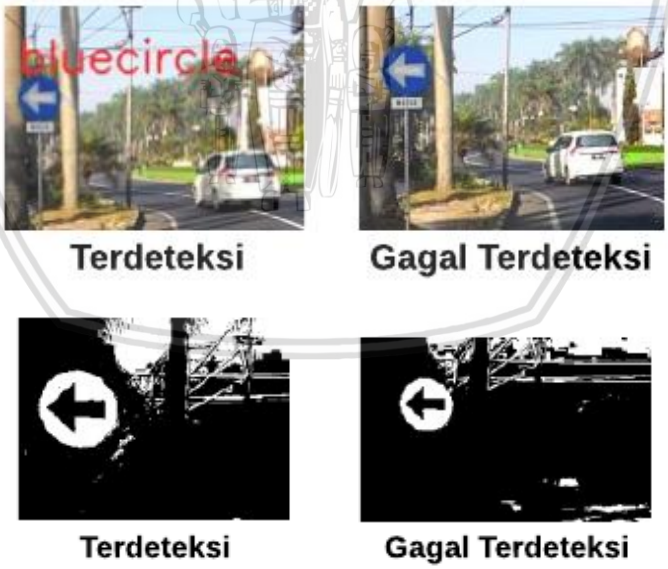
b. Terdeteksi Bentuk
   Saat warna tidak terdeteksi maka otomatis bentuk tidak akan mengenali bentuk. Namun saat warna dikenali maka ada dua kemungkinan yaitu bentuk dari kontur merupakan salah satu bentuk rambu atau tidak sama sekali (*undetected*) sesuai dengan Gamabr 6.7.



Gambar 6.7 Deteksi Bentuk

c. Terdeteksi Jenis

Idetentifikasi warna benar dan deteksi bentuk yang sesuai maka akan menemukan jenis yang benar, *blue-circle* maka disebut dengan rambu perintah. Jika salah satu dari kedua parameter tersebut tidak terpenuhi maka rambu tidak terdeteksi seperti Gambar 6.8.



Gambar 6.8 Deteksi Jenis

Hasil deteksi akan dimasukkan kedalam tabel dengan penjabaran, kolom hasil deteksi merupakan *output* pada *frame* saat berhasil terdeteksi. kolom warna, bentuk, dan jenis menjelaskan jumlah *frame* saat berhasil terdeteksi ketiganya. Dan kolom total adalah kolom yang menjelaskan total *frame* yang terekam dari masing-masing percobaan.

6.3.4.1 Rambu Peringatan

Objek pada *frame* akan dikenali sebagai rambu peringatan jika warnanya kuning dan berbentuk *square*. Pada gambar rambu akan muncul kedua parameter (warna dan bentuk) pada *frame* dan terminal seperti hasil deteksi pada Tabel 6.3.

Tabel 6.3 Deteksi Jenis Rambu Peringatan

Perco-baan	HASIL DETEKSI	Total *Frame*	Warna (*frame*)	Bentuk (*frame*)	Jenis (*frame*)
1		205	158	156	156
2		101	82	82	82
3		30	22	22	22
4		35	31	29	29
5		105	84	84	84
6		75	65	65	65

Perco -baan	HASIL DETEKSI	Total *Frame*	Warna (*frame*)	Bentuk (*frame*)	Jenis (*frame*)
7		125	104	104	104
8		147	144	120	120
9		180	178	138	138
10		31	29	26	26

6.3.4.2 Rambu Perintah

Pada *frame* jika tepat diatas rambu terdeteksi warna rambu perintah (*blue*) artinya warna rambu dikenali, jika tulisan *circle* muncul pada rambu maka bentuk juga berhasil dikenali. Jika kedua parameter benar adanya pada *frame* maka kontur tersebut dapat disimpulkan sebagai rambu perintah. Hasil pendeteksian rambu perintah tertera pada Tabel 6.4.

Tabel 6.4 Deteksi Jenis Rambu Perintah

Perco -baan	HASIL DETEKSI	Total *Frame*	Warna (*frame*)	Bentuk (*frame*)	Jenis (*frame*)
1		64	55	59	59

Perco -baan	HASIL DETEKSI	Total *Frame*	Warna (*frame*)	Bentuk (*frame*)	Jenis (*frame*)
2		82	68	65	65
3		184	173	139	139
4		74	70	63	63
5		95	88	83	83
6		59	48	44	44
7		43	38	36	36

Perco -baan	HASIL DETEKSI	Total *Frame*	Warna (*frame*)	Bentuk (*frame*)	Jenis (*frame*)
8		281	208	208	208
9		69	61	59	59
10		174	147	147	147

6.3.4.3 Rambu Larangan

Saat teridentifikasi warna merah dengan bentuk circle, oktagonal, atau triangle dan sesuai dengan kenyataan pada *frame* maka dapat dinyatakan bahwa objek tersebut merupakan salah satu rambu larangan. Tabel 6.5 menjabarkan hasil deteksi dari rambu larangan.

Tabel 6.5 Deteksi Jenis Rambu Larangan

Perco -baan	HASIL DETEKSI	Total *Frame*	Warna (*frame*)	Bentuk (*frame*)	Jenis (*frame*)
1		75	66	65	65
2		55	47	46	46

Perco -baan	HASIL DETEKSI	Total *Frame*	Warna (*frame*)	Bentuk (*frame*)	Jenis (*frame*)
3		48	38	37	37
4		41	34	30	30
5		89	68	67	67
6		78	65	58	58
7		82	70	68	68
8		73	68	62	62
9		48	40	40	40

Perco -baan	HASIL DETEKSI	Total *Frame*	Warna (*frame*)	Bentuk (*frame*)	Jenis (*frame*)
10		114	90	89	89

Dari pengujian warna rambu, bentuk, dan jenisnya maka ditemukan akurasi dari masing-masing parameter untuk setiap percobaan menggunakan persamaan 6.1.

$$persentase\ percobaan = \frac{frame\ terdeteksi\ (warna,bentuk,atau\ jenis)}{total\ frame}\ x\ 100\% \quad (6.1)$$

Total akurasi dari seluruh percobaan pengujian didapat berdasarkan persamaan 6.2 untuk tiap rambu.

$$persentase\ total = \frac{persentase\ percobaan\ (warna,bentuk,atau\ jenis)}{total\ percobaan} \quad (6.2)$$

6.3.4.4 Rambu Peringatan

Berdasarkan Tabel 6.6 maka diperoleh total akurasi rambu peringatan warna sebesar 86.06%, bentuk 80.55%, dan jenis 80.55%.

Tabel 6.6 Persentase Percobaan Rambu Peringatan

Percobaan	Akurasi Warna(%)	Akurasi Bentuk(%)	Akurasi Jenis(%)
1	77.1	76.1	76.1
2	81.2	81.2	81.2
3	73.33	73.33	73.33
4	88.57	82.86	82.86
5	80	80	80
6	86.67	86.67	86.67
7	83.2	83.2	83.2
8	97.95	81.63	81.63
9	98.89	76.67	76.67
10	96.77	83.87	83,87
Rata-rata Akurasi	**86.06**	**80.55**	**80.55**

6.3.4.5 Rambu Perintah

Total keakuratan pengujian warna dari rambu perintah yaitu 86.67%, bentuk 81.4%, dan jenis rambu 81.4% dari Tabel 6.7.

Tabel 6.7 Persentasi Percobaan Rambu Perintah

Percobaan	Akurasi Warna(%)	Akurasi Bentuk(%)	Akurasi Jenis(%)
1	85.94	84.37	84.37
2	82.93	79.27	79.27
3	94.02	75.54	75.54
4	94.59	85.14	85.14
5	92.63	87.37	87.37
6	81.36	74.58	74.58
7	88.37	83.72	83.72
8	74.02	74.02	74.02
9	88.40	85.51	85.51
10	84.48	84.48	84.48
Rata-rata Akurasi	**86.67**	**81.4**	**81.4**

6.3.4.6 Rambu Larangan

Total persentase ketepatan untuk rambu larangan masing-masing 83.61%(warna), 80.15%(bentuk), dan 80.15%(jenis) yang dijabarkan pada Tabel 6.8.

Tabel 6.8 Persentasi Percobaan Rambu Larangan

Percobaan	Akurasi Warna(%)	Akurasi Bentuk(%)	Akurasi Jenis(%)
1	88	86.67	86.67
2	85.46	83.64	86.64
3	79.17	77.08	77.08
4	82.93	73.17	73.17
5	76.4	75.28	75.28
6	83.33	74.4	74.4
7	85.36	82.92	82.92
8	93.15	84.93	83.93

Percobaan	Akurasi Warna(%)	Akurasi Bentuk(%)	Akurasi Jenis(%)
9	83.33	83.33	83.33
10	78.95	78.07	78.07
Rata-rata Akurasi	**83.61**	**80.15**	**80.15**

6.3.5 Analisis Hasil Pengujian

Setiap frame mendapat kondisi yang berbeda-beda dari lingkungan yaitu kondisi terdeteksi dan gagal terdeteksi. Berikut analisis penyebab keberhasilan dan kegagalan deteksi pada rambu.

a. Terdeteksi
   Saat rambu terdeteksi artinya objek gambar tersebut memenuhi seluruh syarat dan berada pada *range* yang telah ditetapkan pada program. Syarat-syarat tersebut adalah nilai *range* HSV, batas luas kontur, dan jumlah titik puncak yang sesuai dengan program dan juga aspek rasio. dengan terpenuhinya keseluruhan syarat tersebut dengan begitu warna dan bentuk pun ikut terdefenisi baik dengan begitu jenis rambu akan dikenali.
b. Gagal Terdeteksi
   Banyak hal yang mempengaruhi keluaran gagal pada output sistem, yaitu kondisi cahaya yang tidak menentu juga menghambat pendeteksian rambu seperti paparan cahaya yang membuat warna rambu memudar, bayangan dari pohon atau benda lainnya yang menimpa daun rambu membuat warna tidak sesuai dengan range dari ruang warna HSV. Cuaca yang tidak bisa diprediksi seperti mendung atau hujan juga menjadi penghambat pengenalan jenis dari rambu jalan. Sistem yang diimplementasikan pada kendaraan mengharuskan perubahan posisi pendeteksian, pergerakan kendaraan terkadang menghilangkan fokus pada webcam sehingga tidak dapat mendeteksi rambu lalu lintas.

6.4 Pengujian Waktu Deteksi

6.4.1 Tujuan Pengujian

Perhitungan waktu proses deteksi bertujuan menemukan rata-rata waktu yang dibutuhkan untuk mendeteksi rambu. jumlah waktu ini menentukan seberapa baik performa berdasarkan waktu.

6.4.2 Pelaksanaan Pengujian

Waktu proses deteksi menggunakan *library* time dari python. Baris program diletakkan diawal dan akhir proses deteksi.

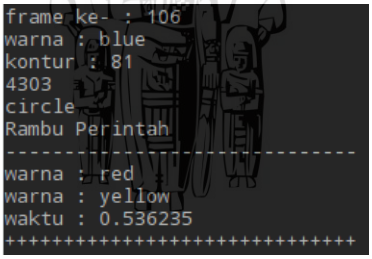
6.4.3 Prosedur Pengujian

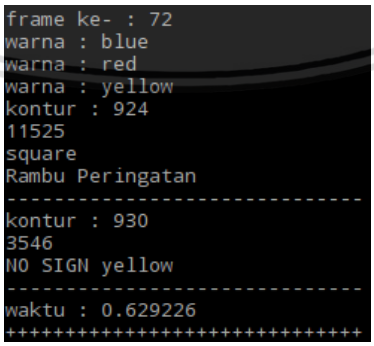
Pengujian waktu deteksi dilakukan dengan prosedur berikut.

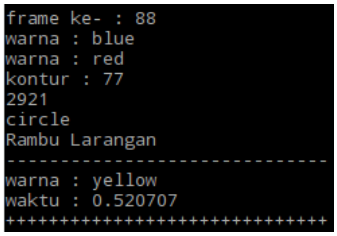
1. Memasang perangkat webcam dengan Raspberry Pi.
2. Menghubungkan Raspberry Pi dengan aplikasi VNC untuk memonitoring Rasp-Pi menggunakan kabel lan.
3. Mengaktifkan tampilan GUI Raspberry Pi pada VNC.
4. Membuka terminal dan menambahkan "import time" pada program deteksi rambu untuk mengakses library waktu pada perangkat.
5. Menambahkan baris program "starttime = time.clock()" pada awal pengolahan *frame* dan "endtime = time.clock()" pada akhir klasifikasi rambu
6. Setelah mendapatkan waktu awal dan akhir, hasil selisih keduanya "waktu = endtime - starttime" diletakkan diakhir setelah "endtime" . kemudian program disimpan kembali.
7. Menjalakan program deteksi dengan "deteksi.py", selama program mendeteksi adanya rambu dalam jarak 2-5 meter maka akan dilakukan perhitungan waktu deteksi tersebut.
8. Mengamati nilai waktu tiap *frame* kemudian mencari rata-rata dari tiap percobaan dan terakhir mencari rata-rata kesuluruhan dari proses deteksi.

6.4.4 Hasil Pengujian

Berikut hasil perhitungan waktu awal, waktu akhir, dan selisih *endtime* dan *starttime* saat rambu terdeteksi dari setiap *frame* tertera seperti Gambar 6.9.







Gambar 6.9 Waktu Deteksi

6.4.5 Analisis Hasil Pengujian

Setelah mendapatkan waktu dari tiap *frame* maka seluruh waktu kemudian dirata-rata lalu dibagi banyak *frame* sehingga diperoleh rata-rata dari tiap percobaan seperti pada Tabel 6.9.

Tabel 6.9 Waktu Deteksi Rambu

Percobaan	Rambu Peringatan (detik)	Rambu Perintah (detik)	Rambu Larangan (detik)
1	0.41	0.40	0.47
2	0.40	0.50	0.48
3	0.87	0.48	0.49
4	0.84	0.59	0.45
5	0.83	0.45	0.49
6	0.4	0.36	0.6
7	0.5	0.49	0.4
8	0.40	0.48	0.48
9	0.82	0.41	0.4
10	0.5	0.4	0.57
Rata-rata waktu	**0.59**	**0.45**	**0.48**

Berdasarkan kesuluruhan waktu percobaan dibagi dengan banyaknya percobaan maka waktu saat terdeteksi rambu pada sistem ini mencapai 0.5 detik atau dengan kata lain 2 *frame* setiap detiknya.

6.5 Pengujian *Output* Suara

6.5.1 Tujuan Pengujian

Output suara dari speaker perlu diuji untuk melihat bagaimana performa program espeak menjalankan fungsi *text to speech*.

6.5.2 Pelaksanaan Pengujian

Speaker telah tersambung dengan port 2.0 menggunakan kabel USB speaker dan kabel jack audio speaker tersambung dengan port audio jack Raspberry Pi.

6.5.3 Prosedur Pengujian

Pengujian output suara pada speaker dilakukakn melalui beberapa tahapan berikut :

1. Mempersiapkan aplikasi VNC dan sudah tersambung dengan Raspberry Pi melalui kabel lan.
2. Menyambungkan antara *speaker* dan port USB Raspberry Pi menggunakan kabel USB.
3. Lampu Led akan aktif berwarna hijau yang artinya *speaker* siap digunakan.
4. Menjalankan program “deteksi.py” .
5. Arahkan webcam ke arah rambu lalu tunggu hingga rambu terdeteksi.
6. Rambu terdeteksi akan menghasilkan suara sesuai rambu yang dideteksi.
7. Memperhatikan keluaran suara pada speaker

6.5.4 Hasil Pengujian

Tanda jika speaker dapat berfungsi adalah led hijau yang menyala seperti [ada Gambar 6.10.

Gambar 6.10 *Speaker* Menyala

Suara yang keluar pada speaker selalu sesuai dengan keluaran terminal. Setelah melakukan percobaan sebanyak sepuluh kali maka didapat hasil sebagai berikut.

Tabel 6.10 Tabel Pengujian Pembacaan Video

Percobaan	Status
1	Berhasil
2	Berhasil
3	Berhasil
4	Berhasil
5	Berhasil
6	Berhasil
7	Berhasil
8	Berhasil
9	Berhasil
10	Berhasil

6.5.5 Analisis Pengujian

Suara yang akan dikeluarkan didukung oleh modul *text to speech* (espeak) jadi untuk setiap keluaran yang sesuai dengan klasifikasi pada program maka akan memberi keluaran yang sesuai juga. Selain kondisi pada klasifikasi tidak akan memberikan *output* suara apa pun sehingga dapat disimpulkan bahwa pengujian ini berhasil seluruhnya.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan maka sebagai jawaban dari rumusan masalah sebelumnya diperoleh tiga kesimpulan sebagai berikut:

1. Sistem deteksi rambu lalu lintas pada Raspberry Pi menggunakan pustaka OpenCV dan bahasa pemrograman python berhasil dirancang melalui tahap perancangan ekstraksi dari video dan *pre-processing* (segmentasi warna, *shape detection*, dan klasifikasi rambu).
2. Gambar biner (*region of interest*) hasil dari segmentasi warna kemudian memasuki tahap pengenalan bentuk (*shape detection*). Metode ini dibagi menjadi tiga bagian yaitu *approximation*, aspek rasio, dan perbandingan luas. fungsi *approximation* berguna untuk menemukan titik pada kontur dimana untuk setiap nilai *approx* sebanyak 3 disebut dengan segitiga (*triangle*), dst. ketentuan kedua adalah perbandingan luas area objek kontur dengan *bounding rectangle* yang mengeliling rambu. ketiga adalah perbandingan panjang dan lebar objek disebut juga dengan aspek rasio.
3. Waktu yang dibutuhkan untuk mendeteksi rambu rata-rata 0.5 detik atau dapat mengolah 2 *frame* perdetiknya.
4. Membedakan rambu dengan objek lainnya disekitarnya dapat menggunakan dua parameter yaitu warna dan bentuk. Maka dari hasil pengujian diperoleh hasil persentase deteksi warna dari rambu lalu lintas (perintah, peringatan, dan larangan) yaitu 85,45%, persentase total dari deteksi bentuk adalah 80.% dan tingkat keberhasilan dari deteksi jenis dari ketiga rambu yaitu 80.7%. dengan begitu berdasarkan hasil pengujian tersebut maka sistem ini sudah cukup baik untuk mendeteksi rambu.

7.2 Saran

Beberapa saran untuk pengembangan lanjut untuk penelitian sejenis adalah sebagai berikut:

1. Pengembangan sistem deteksi rambu pada Raspberry Pi yaitu sistem dapat mendeteksi dari jarak lebih dari 5 meter.
2. Pengembangan dari sistem deteksi pada rambu dapat dilakukan dengan menambahkan metode yang lebih baik untuk klasifikasi jenis rambu yang lebih baik.
3. Penambahan jenis, warna, dan bentuk rambu lainnya untuk dideteksi.
4. Melanjutkan proses deteksi ke tahap pengenalan rambu lalu lintas.

DAFTAR PUSTAKA

Anbarjafari, G. (2014). Digital Image Processing. University Of Tartu.

Chhaya, S. V. (2015). BASIC GEOMETRIC SHAPE AND PRIMARY COLOUR DETECTION 'USING IMAGE PROCESSING ON MATLAB. 5.

E. Petlenkov, A. T. (2015). Raspberry Pi based System for Visual Object Detection and Tracking. Tallin: hgpu.

I. Sebanja, D. M. (2010). Automatic Detection and Recognition of Traffic Road Signs for Intelligent Autonomous Unmanned Vehicles for Urban Surveillance and Rescue. 1-7.

Ilmuti.org. (n.d.). *Ilmuti.org.* Retrieved September 7, 2017, from http://ilmuti.org/wp-content/uploads/2017/05/Tavip-Prahasta_CaraKerjaSistemAutopilotPadaMobil.pdf

Kementerian Perhubungan. (2014, April 14). *Peraturan Menteri Perhubungan Nomor PM 13 Tahun 2014.* Retrieved September 2018, from http://jdih.dephub.go.id/produk_hukum/view/VUUwZ01UTWdWRUZJVlU0Z01qQXhOQT09

Logitech. (n.d.). *Compact Stereo Speakers*. Retrieved from Logitech: https://www.logitech.com/en-roeu/product/stereo-speakers-z120

OpenCV. (2018). *Contour : Getting Started*. Retrieved May 15, 2018, from https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours

OpenCV. (2018). *Morphological Transformation*. Retrieved May 14, 2018, from https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html

OpenCV. (n.d.). *Contours : Getting Started*. Retrieved from Open Source Computer Vision: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

Opencv dev. (2014). *Introduction to OpenCV-Python Tutorials*. Retrieved september 2017, from https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_setup/py_intro/py_intro.html#intro

OpenCV dev. (2018). *Find Contours in your image*. Retrieved May 15, 2018, from https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

opensource.com. (n.d.). *What is Raspberry Pi*. Retrieved oktober 2017, from https://opensource.com/resources/raspberry-pi

P. Shopa, N. S. (2015). Traffic Sign Detection and Recognition Using OpenCV. 1-6.

Puri, R. (2018). Contour, Shape & Color Detection using OpenCV-Python. 3.

S.S.M. Sallah, F. H. (2014). Road Sign Detection and Recoginition System for Real-Time Embedded Applications. *ResearchGate*, 1-5.

Sheikh, M. A. (2016). Traffic Sign Detection and Classification using Colour Feature and Neural Network. 5.